软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(9):2374−2387 [doi: 10.3724/SP.J.1001.2012.04149]

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

求解约束优化问题的ε-DE 算法*

郑建国, 王 翔+, 刘荣辉

(东华大学 旭日工商管理学院,上海 200051)

ε-Differential Evolution Algorithm for Constrained Optimization Problems

ZHENG Jian-Guo, WANG Xiang+, LIU Rong-Hui

(Glorious Sun School of Business and Management, Donghua University, Shanghai 200051, China)

+ Corresponding author: E-mail: shanghaiwx@live.cn

Zheng JG, Wang X, Liu RH. ε-Differential evolution algorithm for constrained optimization problems. *Journal of Software*, 2012,23(9):2374−2387 (in Chinese). http://www.jos.org.cn/1000-9825/4149.htm

Abstract: Differential evolution algorithm usually solves the constrained optimization problems by the feasible solutions priority rule, but the method can not use the infeasible solutions information populations. ε-DE algorithm is designed and can use the information of infeasible solutions. By designing new comparison rules, the infeasible solutions with better objective function are made full use of in the evolution process. The concept of population constraint relax degree is introduced in the comparison rules. During the evolution initial phase, the infeasible solutions with better objective function and near the boundary of the feasible region are incorporated in the population. With the evolutionary generation increasing, the decrease in the population constraint relax degree decreases the number of infeasible solutions in the population. Unless the population constraint relax degree is 0, the population is entirely composed of feasible solutions. In addition, an improved DE algorithm is chosen as the search algorithm, so a faster convergence is gotten. The simulation results of 13 benchmark functions prove that ε-DE is most competitive in all DE algorithms for solving COPs.

Key words: constrained optimization problem; differential evolution algorithm; ε-differential evolution algorithm

摘 要: 差分进化(differential evolution,简称 DE)算法解决约束优化问题(constrained optimization problems,简称 COPs)时通常采用可行解优先的比较规则,但是该方法不能利用种群中不可行解的信息.设计了可以利用不可行解信息的ε-DE 算法.该算法通过构造一种比较准则,使得进化过程可以充分利用种群中优秀不可行解的信息.该准则通过引入种群约束允许放松程度的概念,在进化初始阶段使可行域边界上且拥有较优目标函数的不可行解进入种群;随着进化代数增加,种群约束允许放松程度不断减小,使得种群中不可行解数量减少,直到种群约束允许放松程度为 0,种群完全由可行解组成.此外,还选择了一种改进的 DE 算法作为搜索算法,使得进化过程具有较快的收敛性.13 个标准 Benchmark 函数实验仿真的结果表明:ε-DE 算法是目前利用 DE 算法解决 COPs 问题中效果最好的.

关键词: 约束优化问题;差分进化算法;ε-差分进化算法

中图法分类号: TP18 **文献标识码**: A

* 基金项目: 国家自然科学基金(70971020)
收稿时间: 2010-07-29; 定稿时间: 2011-11-03

差分进化算法(differential evolution,简称 DE)[1]在 IEEE 第 1 届进化计算大赛中表现优异,被广泛应用于各类优化问题.然而,它主要解决全局性优化问题,缺少在受限空间搜索的机制.为了有效解决约束优化问题(constrained optimization problems,简称 COPs),最通常的方法是改进 DE 算法的选择机制,引导种群朝着可行域进化.

Deb 比较准则[2]是最有效的改进 DE 算法选择机制的方法,它由 3 条准则构成:

准则 1. 两个解决方案都是可行解时,拥有较小目标函数的获胜(本文讨论的约束优化问题都是最小化问题).

准则 2. 两个解决方案一个是可行解、一个是不可行解时,可行解获胜.

准则 3. 两个解决方案都是不可行解,约束违反程度之和小的获胜.

Deb 比较准则存在一个明显缺陷,它强调可行解优于不可行解.若按照 Deb 比较准则,则种群进化过程只能在可行域内进行.然而,王勇[3]和 Cai[4]都指出:在解决 COPs 问题时,在种群中保持适当数目的不可行解是找到全局性最优解的关键.

本文提出一种新颖的ε-DE 算法解决 COPs 问题.为了有效地利用不可行解信息,在新算法中设计了一种改进比较机制,从而在种群中引入了接近可行域边界的不可行解,故可以引导搜索过程从可行域和不可行域双侧靠近最优解.该改进比较机制修改了 Deb 准则中的第 2 条准则,其基本思想包含两个方面:第一,使一部分接近可行域边界的不可行解拥有战胜可行解的可能;第二,随着进化代数的增加,这部分不可行解提供的信息越来越少.本文探讨将改进比较机制引入 DE 算法能否有效解决 COPs 问题的 13 个 Benchmark 函数[5];探讨改进比较机制的参数对于解决 COPs 问题的影响.

1 问题定义

COPs 问题的一般形式如下:

$$\min f(X), X \in D \quad \text{s.t.} \begin{cases} g_i(X) \leqslant 0, & i=1,2,\ldots,q \\ h_k(X)=0, & k=q+1,q+2,\ldots,m \end{cases} \tag{1}$$

其中,$D=\prod_{i=1}^{n}[a_i,b_i] \subseteq R^n$,$X=(x_1,x_2,\ldots,x_n)$且 $x_i \in [a_i,b_i]$,$f:D \to R$ 称为目标函数,$g_i(X)$是 q 个不等式约束条件,$h_k(X)$是 $m-q$ 个等式约束条件.同时,令 $S=\{X|X \in D \wedge g_i(X) \leqslant 0 \wedge h_k(X)=0\}$表示可行域.

2 DE 算法及其解决 COPs 问题文献回顾

Price 和 Storn 提出的 DE 算法[1]与遗传算法(genetic algorithm,简称 GA)类似,也包含交叉、变异和选择这 3 个算子,其中最重要是变异算子,该算子的变异信息是种群中随机选择的两个个体的差分信息.DE 算法与 GA 算法的不同之处在于,DE 算法基于实数编码,而 GA 基于二进制编码.

本文选择的 DE 算法伪码如图 1 所示.与标准 DE 算法的不同之处用→标示.*randint*(min,max)产生一个 min 和 max 之间均匀分布的整数.*rand*(0,1)产生一个 0 和 1 之间均匀分布的随机数.*PopulationSize* 表示种群大小,*F* 表示变异的缩放因子,*CR* 表示交叉概率,*n* 表示解决问题的维度.

该算法是 Price[6]在 1997 年为了加快收敛速度提出的一种改进 DE 算法.该算法的变异算子与基本 DE 算法不同,它不是随机选择 X_{r_1} 向量,而是要求 X_{r_1} 向量优于原始向量 X_i,引导变异朝着比 X_i 更优秀的方向进行.所以,该算法比基本 DE 算法收敛速度更快.Price[6]还指出,该改进 DE 算法相比其他进化算法具有以下优势:第一,由于差分向量概率分布均值为 0,进而使得差分变异候选解 X_i' 不存在抽样偏差,故保持了种群多样性;第二,差分向量概率分布标准差随着搜索空间内种群大小和形状改变,因此,处理不同问题时不存在参数敏感性;第三,因为差分向量最终收敛到 0,所以该算法可作为局部搜索算法.综上所述,该算法既能维持种群的多样性又能进行局部搜索,同时收敛速度较快,故本文选择它作为搜索算法.

```
Begin
  gen=0
  Generate initial population {Xi|1≤i≤PopulationSize}
  For gen=1 to MAX_GENERATIONS Do
    For i=1 to PopulationSize Do
→     Select randomly r1≠r2≠r3≠i with r1,r2,r3∈[1,PopulationSize] and  f(X_r1) ≤f(Xi)
      jrand=randint(1,n)
      For j=1 to n Do
        If (rand(0,1)<CR or j=jrand) Then
→         x'_{i,j} = (F + 0.5) * x_{r1,j} + (F − 0.5) * x_{i,j} + F * (x_{r2,j} − x_{r3,j})
        Else
          x'_{i,j} = x_{i,j}
        End If
      End For
      If  f(X'_i) ≤ f(X_i)  Then
        X_i = X'_i
      End If
    End For
  End For
End
```

Fig.1 The pseudo code of improved differential evolution algorithm

图 1 改进差分进化算法伪码

2.1 DE进化算法解决COPs问题文献回顾

Storn[7]提供了一种利用 DE 算法来解决 COPs 的方案.该方案首先对约束进行放松,使得初始种群中所有解都变成可行解.随后,在进化的每一代中,约束条件放松程度不断收紧,同时要求种群中所有个体仍然保持可行解.这个过程不断迭代,一直到约束放松程度为 0.然而,该算法并不适合处理等式约束.

Lampinen[8]也利用 DE 算法解决 COPs,他从另外一个角度处理约束条件.为了改进差分进化算法的选择算子,他设计了以下比较准则:

准则 1. 如果两个个体都是可行解,那么目标函数较优的获胜.

准则 2. 如果一个个体是可行解,另一个是非可行解,那么可行解获胜.

准则 3. 如果两个个体都是非可行解,且在所有约束条件上新产生个体的违反程度都比原始个体小,那么替换原始个体.

该准则与 Deb 准则在准则 3 处存在明显不同:当两个不可行解比较时,Lampinen 提出的算法利用 Pareto 占优概念;而 Deb 准则利用了所有约束违反程度之和.这两种比较准则都存在可行解占优的缺点,当处理最优解位于可行域边界的问题时,这两种准则效果不佳.

Lin[9]使用混合 DE 算法来解决 COPs.他在标准 DE 算法中设计了一些新算子处理种群多样性过大或者过小的问题.在处理约束时,该算法利用了拉格朗日法,将原始目标函数和约束条件合起来构造一个新的目标函数,进而使约束优化问题转化为无约束优化问题.

Mezura[10]也提出了一种 DE 算法解决 COPs.为了增强 DE 算法的搜索能力,该算法允许种群中一个父节点产生多个子孙节点;为了能够处理约束条件,它改进了 DE 算法选择算子,基于概率决定是使用可行解优先的选择标准,还是使用目标函数优先的选择标准.选择算子的改变使得种群保持一定数量不可行解,从而维持了种群多样性.

3 ε-DE 算法

3.1 ε-DE算法设计目标

与全局性优化问题不同,COPs 问题将搜索空间分成可行域和非可行域.根据这个特点,本文提出了ε-DE 算法解决 COPs,新算法存在两个设计目标:1) 找到或接近可行域;2) 找到全局性最优解.

目标 1)主要针对可行域占整个搜索空间比例非常小的情形以及最优解位于可行域边界上的情形.当可行域非常小时,ε-DE 算法期望利用种群中不可行解找到可行域;当最优解位于可行域边界时,它期望同时从可行域和不可行域两侧接近最优解.无论哪种情形,高效利用种群中的不可行解都是ε-DE 算法设计重点.

为了实现目标 2),ε-DE 算法期望设计一种高效的比较选择机制,使得整个进化过程朝着全局性最优解方向进行.

3.2 ε-DE算法两解决方案比较原则

3.2.1 基本概念

定义 1(约束违反程度之和 $G(X)$). $G(X)$表示种群一个个体 X 的约束违反程度之和,其形式如下:

$$G(X)=\sum_{j=1}^{m}G_j(X) \tag{2}$$

其中,

$$G_j(X)=\begin{cases}\max\{0,g_j(X)\}, & 1\leqslant j\leqslant q\\ \max\{0,|h_j(X)|-\delta\}, & q+1\leqslant j\leqslant m\end{cases} \tag{3}$$

其中,δ表示等式约束的放松程度,它是进化代数 gen 的函数:

$$\delta(gen)=\begin{cases}\delta(gen-1)/\theta_1, & \delta>10^{-5}\\ 10^{-5}, & \delta\leqslant 10^{-5}\end{cases} \tag{4}$$

其中,θ_1 表示每进化一代等式约束放松程度δ缩小的比例,取 1.035.

定义 2(种群约束允许放松程度$\varepsilon(gen)$). $\varepsilon(gen)$表示种群进化到 gen 代时个体 X 违反约束程度 $G(X)$的界限,其形式如下:

$$\varepsilon(gen)=\begin{cases}\varepsilon(gen-1)/\theta_2, & \varepsilon>10^{-6}\\ 0, & \varepsilon\leqslant 10^{-6}\end{cases} \tag{5}$$

其中,θ_2 表示每进化 1 代种群约束允许放松程度减小的比例,取 1.035.

定义 3(可接受解). 当种群进化到第 gen 代时,如果 $0<G(X)\leqslant\varepsilon(gen)$,个体 X 就是可接受解.

定义 4(不可接受解). 当种群进化到第 gen 代时,如果 $G(X)>\varepsilon(gen)$,个体 X 就是不可接受解.

3.2.2 两个解决方案比较规则

为了实现ε-DE 算法两个设计目标,利用定义 1~定义 4 设计了两解决方案比较的 3 准则.

准则 1. 两个都是可行解,目标函数小的获胜.

准则 2. 两个都是不可行解,约束违反程度 $G(X)$小的获胜.

准则 3. 一个是可行解一个是不可行解,分两种情形讨论:

情形 1:如果不可行解是可接受解,那么比较不可行解和可行解的目标函数,小的获胜;

情形 2:如果不可行解是不可接受解,那么可行解获胜.

将以上比较准则定义成两个解决方案的比较函数 $Prior(X_1,X_2)$.

定义 5(两解决方案比较函数 $Prior(X_1,X_2)$).

$$Prior(X_1,X_2)=\begin{cases}f(X_1)\leqslant f(X_2), & G(X_1)=G(X_2)=0\\ G(X_1)\leqslant G(X_2), & G(X_1)>0\wedge G(X_2)>0\\ f(X_1)\leqslant f(X_2), & G(X_1)=0\wedge\varepsilon(gen)\geqslant G(X_2)>0\\ 1, & G(X_1)=0\wedge G(X_2)>\varepsilon(gen)\\ f(X_1)\leqslant f(X_2), & \varepsilon(gen)\geqslant G(X_1)>0\wedge G(X_2)=0\\ 0, & G(X_1)>\varepsilon(gen)\wedge G(X_2)=0\end{cases} \tag{6}$$

函数 $Prior(X_1,X_2)$返回值为 1 表示 X_1 优于 X_2,返回值为 0 表示 X_1 劣于 X_2.

这个比较准则与 Deb 比较准则不同,差异表现在两个方面:第一,基本思想不一样.Deb 比较准则基于可行解优先的思想,ε-DE 算法比较准则基于充分利用可行域边界附近不可行解信息的思想;第二,当两比较方案一个

是可行解,一个是不可行解时比较准则不同,Deb 比较准则可行解一定获胜,而在ε-DE 算法的比较准则中,若不可行解是可接受解且具有较优的目标函数,则可行解失败.

3.3 ε-DE算法框架

ε-DE 算法基于第 3 节介绍的改进 DE 算法,并将新设计的三比较规则函数 $Prior(X_1,X_2)$嵌套其中,以下是该算法框架的伪码(如图 2 所示).与图 1 中改进 DE 算法不同之处用→标示出来.*randint*(min,max)产生一个 min 和 max 之间均匀分布的整数.*rand*(0,1)产生一个 0 和 1 之间均匀分布的随机数.*PopulationSize* 表示种群大小,*F* 表示变异的缩放因子,*CR* 表示交叉概率,*n* 表示解决问题的维度.

Begin
  gen=0
  Generate initial population $\{X_i|1\leqslant i\leqslant PopulationSize\}$
  For *gen*=1 to *MAX_GENERATIONS* **Do**
    For *i*=1 to *PopulationSize* **Do**
→   Select randomly $r_1\neq r_2\neq r_3\neq i$ with $r_1,r_2,r_3\in[1,PopulationSize]$ and $Prior(X_{r_1},X_i)=1$
      $j_{rand}=randint(1,n)$
      For *j*=1 to *n* **Do**
        If (*rand*(0,1)<*CR* or $j=j_{rand}$) **Then**
          $x'_{i,j}=(F+0.5)*x_{r_1,j}+(F-0.5)*x_{i,j}+F*(x_{r_2,j}-x_{r_3,j})$
        Else
          $x'_{i,j}=x_{i,j}$
        End If
      End For
→   **If** ($Prior(X'_i,X_i)=1$) **Then**
        $X_i=X'_i$
      End If
    End For
  End For
End

Fig.2 The pseudo code of ε-DE algorithm

图 2 ε-DE 算法实现的伪码

与图 1 中改进 DE 算法相比,ε-DE 算法进行了两处变化,分别用→进行了标示,具体改变表现为利用两解决方案比较规则函数 $Prior(X_1,X_2)$替代了目标函数 $f(X)$的比较.

3.4 算法时间复杂度分析

当度量进化算法的计算复杂度时,目前最常用的方法是以一次适应度函数评价为基本单位,基于此标准,基本 DE 算法的计算复杂度是 $O(MAX\_GENERATIONS\times PopulationSize)$;同时,根据图 2 可知,$\varepsilon$-DE 算法的计算复杂度同样为 $O(MAX\_GENERATIONS\times PopulationSize)$,即两者的计算复杂度完全相同.MAX_*GENERATIONS* 代表最大迭代次数,*PopulationSize* 表示种群尺寸.

事实上,与基本 DE 算法相比,ε-DE 算法在变异算子和选择算子都作出了较大改进,确实会在一定程度上降低算法的运行速度.这主要表现为在进行变异操作时,新算法要求 $Prior(X_{r_1},X_i)=1$.即,需要找到比当前个体 X_i 更优秀的个体 X_{r_1}.为实现该操作,在进化过程的每一代中,必须先将整个种群$\{X_j|j=1,\ldots,PopulationSize\}$按照第 3.2.2 节的改进比较准则进行排序.然而,与适应度函数评价耗费的时间相比,排序操作增加的时间基本可以忽略.这是由于约束优化问题的目标函数和约束条件都比较复杂,计算非常耗时;同时,在进化过程的每一代中,种群中每个个体都需要计算一次目标函数和约束条件.

3.5 讨 论

3.5.1 ε-DE 算法如何利用三比较原则实现设计目标

当处理最优解位于可行域边界的问题时,在可行域边界附近且拥有较优目标函数的不可行解.利用第 3 条

比较准则的情形 1,从而引导整个种群从不可行域一侧靠近最优解.

当处理可行域占整个搜索空间比例太小的问题时,由第 2 条比较准则可知,若不可行解距离可行域边界越近则越优先,故整个种群不断接近可行域.

随着进化代数增加,种群约束允许放松程度$\varepsilon(gen)$不断减小,这使得只有更靠近可行域边界的不可行解才能为进化过程提供有用信息.对于极端情形$\varepsilon(gen)$=0,种群中所有的不可行解都不能为进化过程再提供信息.此时,进化过程演变成为在可行域内的寻优过程.

3.5.2 ε-DE 算法与其他算法的区别

与ε-DE 算法一样,Storn 算法[7]也使用 DE 算法解决 COPs.两者都是基于种群约束放松程度的概念,但是有 3 点区别:第一,约束放松的目的不同.ε-DE 算法约束放松的目的是为了利用不可行解,Storn 算法约束放松的目的是为了寻找可行域;第二,约束收缩的方式不同.ε-DE 算法使用线性收缩,每次收缩程度都相同,Storn 算法依赖种群中的最差解进行收缩;第三,进化方式不同.ε-DE 算法利用改进的 Deb 规则驱动进化过程,Storn 算法利用 ROA 种群边界收缩原则驱动进化过程.

林丹[11]提出的解决 COPs 的 GA 算法同样是基于种群约束放松程度的概念,该算法与ε-DE 算法存在 3 点不同:第一,基本思想不同.林丹算法认为,在整个进化过程中,种群都需要保持一定数量的不可行解,而ε-DE 算法认为随着进化代数增加,不可行解提供的信息越来越少;第二,种群约束放松程度ε改变的算法不同.林丹算法约束放松程度ε自适应变化,而ε-DE 算法的约束放松程度ε随着进化代数增加一直在减小,直到为 0 停止;第三,搜索算法不同.林丹算法采用 GA 作为搜索算法,而ε-DE 算法采用 DE 算法作为搜索算法.

4 实验结果及讨论

为了比较算法的性能,我们使用 Runarsson 和 Yao[5]提供的 13 个 Benchmark 函数对ε-DE 算法进行测试.13 个函数主要特征见表 1.其中,n 表示问题规模,Function 表示目标函数的类型,ρ表示可行域占整个搜索空间的比例,LI 表示线性不等式约束的个数,NI 表示非线性不等式约束的个数,LE 表示线性等式约束的个数,NE 表示非线性等式约束的个数.

Table 1 The characteristic of 13 benchmarck functions

表 1 13 个 Benchmark 函数的特征表

Problem	n	Function	ρ(%)	LI	NI	LE	NE
g01	13	quadratic	0.000 3	9	0	0	0
g02	20	nonlinear	99.997 3	1	1	0	0
g03	10	nonlinear	0.002 6	0	0	0	1
g04	5	quadratic	27.007 9	0	6	0	0
g05	4	nonlinear	0.000 0	2	0	0	3
g06	2	nonlinear	0.005 7	0	2	0	0
g07	10	quadratic	0.000 0	3	5	0	0
g08	2	nonlinear	0.858 1	0	2	0	0
g09	7	nonlinear	0.519 9	0	4	0	0
g10	8	linear	0.002 0	3	3	0	0
g11	2	quadratic	0.097 3	0	0	0	1
g12	3	quadratic	4.769 7	0	93	0	0
g13	5	nonlinear	0.000 0	0	0	0	3

本文在 Windows XP SP3 操作系统下,利用 Matlab 2009a 编码实现了ε-DE 算法.随后,在处理器为 Intel Core 2 Duo 3GHZ、内存为 2GB 的 PC 机上,针对 13 个 Benchmark 函数分别独立执行 50 次实验.实验的算法参数见表 2.其中,Problem 表示问题编号;*PopulationSize* 表示种群规模;*MAX_GENERATIONS* 代表差分进化的最大进化代数;F 表示差分进化的缩放因子;*CR* 表示差分进化算法的交叉概率;ε_0 表示约束容忍程度的初始数值;δ_0 表示等式约束放松程度的初始数值,没有等式约束的 Benchmark 函数不需要设置此项.

利用ε-DE 算法对 13 个 Benchmark 函数进行实验时,大部分实验参数设置完全一致,只有 g02,g13 中与其他 Benchmark 函数有些不同,详细讨论见第 4.4 节.因为ε-DE 算法的参数 *PopulationSize*=200, MAX_*GENERATIONS*=10000,所以目标函数评价次数 *FFEs*=2000000.

Table 2 The experimental parameters of ε-DE algorithm on 13 benchamark functions (NA means null for the parameter)

表 2 针对 13 个 Benchmark 函数ε-DE 算法的实验参数(NA 表示不用设置该参数)

Problem	*PopulationSize*	MAX_*GENERATIONS*	*F*	*CR*	ε_0	δ_0
g01	200	10 000	0.5	0.9	1	NA
g02	200	10 000	0.5	0.9	15	NA
g03	200	10 000	0.5	0.9	1	1
g04	200	10 000	0.5	0.9	1	NA
g05	200	10 000	0.5	0.9	1	1
g06	200	10 000	0.5	0.9	1	NA
g07	200	10 000	0.5	0.9	1	NA
g08	200	10 000	0.5	0.9	1	NA
g09	200	10 000	0.5	0.9	1	NA
g10	200	10 000	0.5	0.9	1	NA
g11	200	10 000	0.5	0.9	1	1
g12	200	10 000	0.5	0.9	1	NA
g13	200	10 000	0.5	0.9	20	20

4.1 本文算法的总体性能

表 3 展示了 13 个 Benchmark 函数 50 次独立实验的统计结果.每个函数都列出了已知最优值、ε-DE 算法 50 次独立实验结果的最优值、中位数、均值、最差值和标准差.

Table 3 The stastical results of ε-DE algorithm in 50 independent experiments on 13 benchmark functions

表 3 ε-DE 算法针对 13 个 Benchmark 函数 50 次独立实验的统计结果

Fcn/Optimal	Best	Median	Mean	Worst	st.dev
g01/−15.000	−15.000	−15.000	−15.000	−15.000	0
g02/−0.803619	−0.803619	−0.803619	−0.803004	−0.792608	2.5E−03
g03/−1.000	−1.000	−1.000	−1.000	−1.000	3.9E−06
g04/−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	2.1E−05
g05/5126.498	5126.498	5126.498	5126.498	5126.498	1.7E−05
g06/−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	2.3E−08
g07/24.306	24.306	24.306	24.306	24.306	6.3E−06
g08/−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	8.4E−17
g09/680.63	680.630	680.630	680.630	680.630	2.2E−07
g10/7049.248	7049.248	7049.248	7049.248	7049.248	9.0E−06
g11/0.75	0.75	0.75	0.75	0.75	6.9E−14
g12/−1.000	−1.000	−1.000	−1.000	−1.000	0
g13/0.053950	0.053949	0.053949	0.069631	0.438846	7.6E−02

由表 3 可知,ε-DE 算法实验结果良好,除函数 g02 和 g13 之外,剩下 11 个 Benchmark 函数 50 次实验都找到了全局性最优解.图 3(a)和图 3(b)描述了 g02 和 g13 的 50 次实验最优解分布情况(直方图上方标示的是该数值处获得最优解的次数).虽然 g2 函数 50 次实验没有完全成功,但是有 47 次达到最优解.g13 函数 50 次实验成功次数也达到了 47 次.此外,通过表 3 还可以看出,13 个函数的标准差都非常小,这表示ε-DE 算法鲁棒性更好.

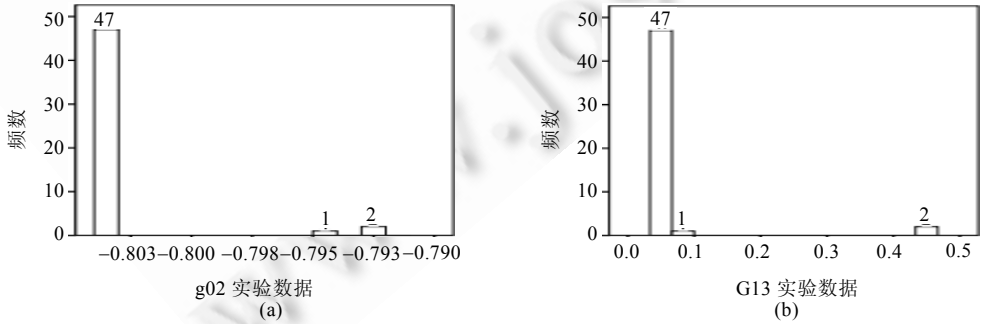

Fig.3 The histograms of the optimal solutions in 50 independent experiments on g02 and g13

图 3 函数 g02 和 g13 函数 50 次独立实验最优解的直方图

表 2 和图 3 的结果显示,ε-DE 算法在没有使用局部搜索算法加强搜索能力的情形下,求解 COPs 仍然具有竞争力.

4.2 本文算法与其他不同约束处理算法的比较

为了验证ε-DE 算法的有效性,我们通过表 4 比较它与其他算法解决 13 个 Benchmark 函数的质量.其他算法包括:

1) The improved version of the Stochastic Ranking (ISR)[12];
2) The Hybrid Differential Evolution (HDE)[9];
3) The Extended Differential Evolution (EXDE)[8];
4) The Diversity Differential Evolution (Diversity-DE)[10].

上述 4 种算法中,IRY 算法[12]改进了经典 RY 算法[5],是目前已知解决 COPs 中最有竞争力的一个.为了与ε-DE 算法具有可比性,其他选择的都是 DE 类算法.

Table 4 The comparison of the stastical results on 13 benchmark functions using ε-DE, ISR,RDE,HDE,EXDE or Diversity-DE

表 4 针对 13 个 Benchmark 函数,ε-DE 算法与 ISR,RDE,HDE,EXDE 和 Diversity-DE 的统计结果比较

Fcn/Optimal	Status	Methods				
		ISR	HDE	EXDE	Diversity-DE	ε-DE
g01/ −15.000	Best	−15.000	−15.000	−15.000	−15.000	−15.000
	Mean	−15.000	−15.000	−15.000	−15.000	−15.000
	Worst	−15.000	−15.000	−15.000	−15.000	−15.000
	st.dev	5.8E-14	NA	NA	1.0E−9	0
g02/ −0.803619	Best	−0.803619	NA	NA	−0.803619	−0.803619
	Mean	−0.782715	NA	NA	−0.798079	−0.803004
	Worst	−0.723591	NA	NA	−0.751742	−0.792608
	st.dev	2.2E-2	NA	NA	1.01E−2	2.47E−03
g03/ −1.000	Best	−1.001	NA	NA	−1.000	−1.000
	Mean	−1.001	NA	NA	−1.000	−1.000
	Worst	−1.001	NA	NA	−1.000	−1.000
	st.dev	**8.2E−9**	NA	NA	**0**	3.9E−06
g04/ −30665.539	Best	−30665.539	NA	NA	−30665.539	−30665.539
	Mean	−30665.539	NA	NA	−30665.539	−30665.539
	Worst	−30665.539	NA	NA	−30665.539	−30665.539
	st.dev	**1.1E−11**	NA	NA	**0**	2.1E−05
g05/ 5126.498	Best	5126.497	NA	5126.484	5126.497	5126.498
	Mean	5126.497	NA	5126.484	5126.497	5126.498
	Worst	5126.497	NA	5126.484	5126.497	5126.498
	st.dev	**7.2E−13**	NA	NA	**0**	1.7E−05
g06/ −6961.814	Best	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814
	Mean	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814
	Worst	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814
	st.dev	**1.9E−12**	NA	NA	**0**	2.3E−08
g07/ 24.306	Best	24.306	24.306	24.306	24.306	24.306
	Mean	24.306	24.306	24.306	24.306	24.306
	Worst	24.306	24.307	24.307	24.306	24.306
	st.dev	6.3E−5	NA	NA	**8.22E−9**	6.3E−06
g08/ −0.095825	Best	−0.095825	NA	−0.095825	−0.095825	−0.095825
	Mean	−0.095825	NA	−0.095825	−0.095825	−0.095825
	Worst	−0.095825	NA	−0.095825	−0.095825	−0.095825
	st.dev	**2.7E−17**	NA	NA	**0**	8.4E−17
g09/ 680.63	Best	680.630	680.630	680.630	680.630	680.63
	Mean	680.630	680.631	680.630	680.630	680.63
	Worst	680.630	680.634	680.630	680.630	680.63
	st.dev	**3.2E−13**	NA	NA	**0**	2.2E−07
g10/ 7049.248	Best	7049.248	7049.862	7049.248	7049.248	7049.248
	Mean	7049.250	7055.079	7049.248	7049.266	7049.248
	Worst	7049.270	7116.188	7049.248	7049.617	7049.248
	st.dev	3.2E−3	NA	NA	4.45E−2	9.0E−06

Table 4 The comparison of the stastical results on 13 benchmark functions using ε-DE, ISR,RDE,HDE,EXDE or Diversity-DE (Continued)

表 4 针对 13 个 Benchmark 函数,ε-DE 算法与 ISR,RDE,HDE,EXDE 和 Diversity-DE 的统计结果比较(续)

Fcn/Optimal	Status	Methods				
		ISR	HDE	EXDE	Diversity-DE	ε-DE
g10/ 7049.248	Best	7049.248	7049.862	7049.248	7049.248	7049.248
	Mean	7049.250	7055.079	7049.248	7049.266	7049.248
	Worst	7049.270	7116.188	7049.248	7049.617	7049.248
	st.dev	3.2E−3	NA	NA	4.45E−2	9.0E−06
g11/ 0.75	Best	0.75	NA	0.75	0.75	0.75
	Mean	0.75	NA	0.75	0.75	0.75
	Worst	0.75	NA	0.75	0.75	0.75
	st.dev	3.2E−3	NA	NA	**0**	6.9E−14
g12/ −1.000	Best	−1.000	NA	NA	−1.000	−1.000
	Mean	−1.000	NA	NA	−1.000	−1.000
	Worst	−1.000	NA	NA	−1.000	−1.000
	st.dev	1.2E−9	NA	NA	0	0
g13/ 0.053950	Best	0.053942	0.053950	NA	0.053941	0.053949
	Mean	**0.06677**	**0.053950**	NA	**0.069336**	0.069631
	Worst	**0.438803**	**0.053950**	NA	**0.438803**	0.438846
	st.dev	**7.0E−2**	NA	NA	**7.58E−2**	7.6E−02

NA 表示数据不能获得.如果前 4 种算法统计结果中优于ε-DE 算法的用粗体表示

由表 4 可知,ε-DE 算法共有 11 个 Benchmark 函数 50 次实验全部达到了目前已知最优解,而其他 4 种算法中最优秀的 ISR 算法和 Diversity-DE 算法都只成功求解了 10 个函数.g02 函数可行域大而且目标函数复杂,5 种算法中有两种没有对它进行实验,剩下 3 种算法都没有在进行的实验中全部找到最优解;但是ε-DE 算法 50 次独立实验只失败了 3 次,而且无论是最差解还是最优解均值都是 3 种算法中最优秀的.g10 函数约束条件复杂,5 种算法中有 3 种在进行的实验中没有全部达到最优解,只有ε-DE 算法和 EXDE 算法全部找到最优解.

ISR 算法[12]是目前解决 COPs 算法中最优秀的,而且不是基于 DE 算法.针对函数 g13,通过比较 ISR 和ε-DE 的均值和最差解,得出 ISR 优于ε-DE;但是考虑到ε-DE 进行了 50 次独立实验,ISR 只执行了 30 次实验,同时两者均值和标准差相差很小,故可认为在处理 g13 函数时两算法能力相当.然而,ε-DE 算法在 g02 和 g10 上的结果明显优于 ISR 算法.

Diversity-DE 算法[10]是目前文献中解决 COPs 中最优秀的改进 DE 算法.通过比较 Diversity-DE 和ε-DE 求解 g13 函数的均值和最差解,得出 Diversity-DE 优于ε-DE;但是考虑到ε-DE 进行了 50 次独立实验,Diversity-DE 执行了 100 次实验,同时两者均值和标准差相差很小,可知在处理 g13 函数时两算法能力相当.然而,ε-DE 算法在 g02 和 g10 上的结果却明显优于 ISR 算法.

通过以上分析可知,解决 COPs 时,无论与非 DE 类算法相比还是与 DE 类算法相比,ε-DE 算法都非常具有竞争力.

4.3 为什么ε-DE算法可以解决COPs问题

为了探索ε-DE 算法解决 COPs 的原理,我们记录了 13 个 Benchmark 函数 50 次实验的进化过程(见表 5).每个函数都列出了 50 次独立实验随着进化代数增加,可行解均值 *feasible_mean*、不可行解均值 *unfeasible_mean*、不可行解距离可行域边界距离的均值 *unfeasible_dis* 和不可行解数目的均值 *unfeasible_num*.

通过指标 *feasible_mean* 可以得出:随着进化代数增加,所有 13 个 Benchmark 函数的可行解均值都在向最优解逼近.因为在 3 个比较准则中,引导可行域内进化过程的是准则 1——目标函数优先原则,所以可行解均值向可行域内最优解逼近.这说明ε-DE 算法从可行域一侧朝最优解逼近.

通过指标 *unfeasible_mean* 和 *unfeasible_dis* 可以得出:随着进化代数的增加,所有 13 个 Benchmark 函数的不可行解均值向可行域最优解方向逼近,不可行解与可行域边界的距离均值不断减小.不可域内的进化过程取决于两个因素:一是ε-DE 算法的 3 个比较准则,二是关于种群约束放松程度ε缩减的公式(5).在这两个因素作用下,无论是可接受解还是不可接受解,都朝着最优解方向逼近,说明ε-DE 算法从不可行域一侧朝着最优解逼近.

Table 5 The evolutionary process of 50 independent experiments on 13 benchmark functions

表 5 13 个 Benchmark 函数 50 次独立实验的进化过程

Test	Status	Generations						
		1	51	101	201	301	401	451
g01/ −15.000	*feasible_mean*	NA	−13.714	−14.874	−14.997	−15	−15	−15
	unfeasible_mean	−150.43	−13.677	−14.868	−14.997	−15	−15	NA
	unfeasible_dis	563.05	0.056182	0.007245	0.00018652	5.90E−06	1.77E−07	NA
	unfeasible_num	200	135.36	153.32	160.26	159.96	161.86	0
g02/ −0.803619	*feasible_mean*	−0.097499	NA	−0.091578	−0.17513	−0.24171	−0.28843	−0.30752
	unfeasible_mean	NA	−0.11818	−0.11445	NA	NA	NA	NA
	unfeasible_dis	NA	0.74986	0.74974	NA	NA	NA	NA
	unfeasible_num	0	198.16	150.12	0	0	0	0
g03/ −1.000	*feasible_mean*	−0.31954	−0.60674	−0.79709	−0.98495	−0.99926	−1	0
	unfeasible_mean	−104.01	−0.78046	−0.83315	−0.9841	−0.99919	−1	0
	unfeasible_dis	1.5065	0.071798	0.014411	0.00056411	2.05E−05	5.30E−07	0
	unfeasible_num	187.24	162.12	145.7	128.32	122.48	76.46	0
g04/ −30665.539	*feasible_mean*	−26 606	−30 573	−30 651	−30 665	−30 666	−30 666	0
	unfeasible_mean	−28 018	−30 575	−30 652	−30 665	−30 666	−30 666	0
	unfeasible_dis	1.3465	0.02464	0.0036485	9.79E−05	3.00E−06	9.73E−08	0
	unfeasible_num	145.46	173.3	173.6	173.58	172.82	175.08	0
g05/ 5126.498	*feasible_mean*	NA	5228.7	5129.5	5126.6	5126.5	5126.5	5126.5
	unfeasible_mean	3728.4	5257.3	5130.4	5126.6	5126.5	5126.5	NA
	unfeasible_dis	1955.8	0.29643	0.013331	0.00030149	8.75E−06	1.57E−07	NA
	unfeasible_num	200	181.76	178.1	189.82	190.92	164.56	0
g06/ −6961.814	*feasible_mean*	NA	−6934.3	−6961.4	−6961.8	−6961.8	−6961.8	0
	unfeasible_mean	2.92E+05	−6943.8	−6961.4	−6961.8	−6961.8	−6961.8	0
	unfeasible_dis	5969.1	0.0082468	0.00019777	1.79E−06	4.60E−08	6.87E−10	0
	unfeasible_num	200	177.76	176.56	172.3	170.72	171.96	0
g07/ 24.306	*feasible_mean*	NA	51.188	27.947	24.57	24.337	24.31	24.308
	unfeasible_mean	2182.4	48.685	27.678	24.548	NA	NA	NA
	unfeasible_dis	2009.3	0.093225	0.01552	0.0005233	NA	NA	NA
	unfeasible_num	200	21.34	28.6	13.5	3.42	1.02	0
g08/ −0.095825	*feasible_mean*	NA	−0.087452	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
	unfeasible_mean	−0.1998	−0.085067	NA	NA	NA	NA	NA
	unfeasible_dis	37.19	0.14179	NA	NA	NA	NA	NA
	unfeasible_num	198.34	42.72	0	0	0	0	0
g09/ 680.63	*feasible_mean*	NA	691.09	680.84	680.63	680.63	680.63	0
	unfeasible_mean	1.44E+06	689.03	680.82	680.64	680.63	680.63	0
	unfeasible_dis	6655.4	0.088612	0.013304	0.00030731	9.34E−06	3.05E−07	0
	unfeasible_num	198.84	13.38	88.64	126.32	128.9	128.34	0
g10/ 7049.248	*feasible_mean*	NA	11203	8483.7	7221.5	7065.4	7050.8	7049.7
	unfeasible_mean	16023	11735	8638.2	7219	7064.3	7050.6	NA
	unfeasible_dis	1.71E+06	0.041387	0.0094566	0.00049839	1.88E−05	6.06E−07	NA
	unfeasible_num	200	141.58	119.54	53.22	19.14	7.36	0
g11/ 0.75	*feasible_mean*	1.2521	NA	NA	NA	NA	0.74999	0
	unfeasible_mean	3.4645	0.5893	0.71883	0.749	0.74997	0.74999	0
	unfeasible_dis	0.30238	0.036892	0.005063	0.00015463	4.99E−06	2.31E−09	0
	unfeasible_num	37.6	195.94	196.42	196.2	195.78	178.9	0
g12/ −1.000	*feasible_mean*	−0.80758	−0.97115	−0.9994	−1	−1	−1	0
	unfeasible_mean	−0.74829	-0.91641	NA	NA	NA	NA	0
	unfeasible_dis	0.35492	0.032709	NA	NA	NA	NA	0
	unfeasible_num	191.2	166.3	6.9	0	0	0	0
g13/ 0.053950	*feasible_mean*	1.41E+21	0.39844	0.29001	NA	NA	NA	NA
	unfeasible_mean	5.48E+40	1.1371	0.49344	0.25803	0.21689	0.20659	0.20479
	unfeasible_dis	9.8917	1.4292	0.58974	0.050164	0.002498	3.00E−05	4.00E−06
	unfeasible_num	51.2	151.16	178.32	181.8	183.24	181.4	145.76

进化代数记录了有代表性的第 1 代、第 51 代、第 101 代、第 201 代、第 301 代、第 401 代和第 451 代.NA 表示数据不能获得.

指标 *unfeasible_num* 的变化情况很复杂,根据该指标的表现形式,将 13 个函数分成 4 类:

- 类型 1:函数 g7,g8,g10,g12 的特征是随着进化代数的增加,种群中不可行解数目一直在减少,直到 0.这 4 个案例的共同特征是最优解都位于可行域内部.由以上论证可知,不可性解的进化方向是可行域内的

最优解,故不可行解会快速突破可行域边界,使得种群完全由可行解组成;

- 类型 2:函数 g1,g3,g4,g5,g6,g9,g11 的特征是随着进化代数的增加,种群中不可行解的数目开始减少,之后稳定在一个数值,最后减少到 0.这 7 个案例的共同特征是最优解都位于可行域边界上.根据上一部分论证,不可行解朝着最优解逼近,因此开始阶段不可行解的数目一直在减少;随后,当不可行解接近可行域边界时,如果种群约束放松程度ε>0,则进化过程在可行域边界区域进行,从而使得不可行解的数目进入一个相对平稳的时期;然而,当进化代数等于 451 次时,ε=0,比较准则演变成可行解优先的 Deb 准则,故不可行解的数目变为 0;
- 类型 3:函数 g2 的特征是种群中不可行解的数目开始为 0,之后增多,随后快速减少到 0.函数 g2 有两个特点:第一是可行域非常大,达到了 99.9973%,因此,初始时种群全部都是可行解;第二个是最优解位于可行域内部,根据类型 1 的结论,不可行解的数目会快速减少到 0,整个进化演变成可行域内进化.由以上分析可知,类型 3 是类型 1 的一个特例;
- 类型 4:函数 g13 表现非常复杂,它是ε-DE 算法没有完全解决的两个问题之一,它的 50 次实验进化情况分成 3 种模式,故统计数据没有表现出一致性(详细的讨论和本文主题无关).

综上所述,ε-DE 有效解决 COPs 的原因有 3 个:

- 最优解无论位于可行域内部还是边界上,ε-DE 算法的三比较准则都能保证从可行域和不可行域两个方向逼近最优解;
- 关于ε缩减的公式(5),保证不可行解在进化开始阶段提供较多的信息;随着进化代数的增加,不可行解为进化提供的信息越来越少;
- ε-DE 算法中,DE 算法是一种高效的寻优算法,主导了整个进化过程.

4.4 参数ε和δ的敏感性分析

ε-DE 算法中,最重要的两个参数是种群约束允许放松程度ε和等式约束放松程度δ.因为只有含有等式约束的函数才有参数δ,所以分成含有等式约束和不含等式约束两种情形讨论.利用不含等式约束的函数讨论ε的参数效应,利用含有等式约束的函数讨论δ的参数效应.

4.4.1 参数ε敏感性分析

表 6 展示了不含等式约束的 Benchmark 函数 g01,g02,g04,g06,g07,g08,g09,g10 和 g12,针对种群约束允许放松程度ε的 6 种不同取值 0.1,0.5,1,5,10,15 和 20,分别进行 50 次独立实验达到最优解的次数.

Table 6 The statistical results of 50 independent experiments with different parameter ε on the benchmark functions with inequality costraints (g01,g02,g04,g06,g07,g08,g09,g10 and g12)

表 6 针对不含等式约束 Benchmark 函数 g01,g02,g04,g06,g07,g08,g09,g10 和 g12,每个参数ε进行 50 次独立实验的统计结果

Fcn/Optimal	Status	Different parameter values						
		ε=0.1	ε=0.5	ε=1	ε=5	ε=10	ε=15	ε=20
g01/	Times	45	50	50	50	50	50	50
g02/	Times	39	38	42	41	36	47	43
g04/	Times	50	50	50	50	50	50	50
g06/	Times	50	50	50	50	50	50	50
g07/	Times	50	50	50	50	50	50	50
g08/	Times	50	50	50	50	50	50	50
g09/	Times	50	50	50	50	50	50	50
g10/	Times	50	50	50	50	50	50	50
g12/	Times	50	50	50	50	50	50	50

如果 50 次实验没有全部成功,用粗体表示.

ε表示可行域边界放松的程度,它使算法能够从可行域和不可行域两侧接近最优解.利用ε可以加大搜索范围,不仅在可行域内搜索,还可以在可行域边界附近的不可行域内搜索.ε越大,表示从可行域边界越远的地方开始搜索,获得最优解的概率也越大.由表 6 中可知,所有案例随着ε的增加,实验结果没有变差.参数ε变小会减少算

法在不可行域内搜索的范围,会减小找到最优解的概率.例如,g01 的ε=0.1 时,50 次实验只成功了 45 次;而当ε>0.1 时,50 次实验全部成功.

4.4.2 参数δ敏感性分析

表 7 展示了含有等式约束的 Benchmark 函数 g03,g05,g11 和 g13,针对等式约束放松程度δ和种群约束允许放松程度ε的 6 种不同取值 0.1,0.5,1,5,10,15 和 20,分别进行 50 次独立实验达到最优解的次数.

Table 7 The statistical results of 50 independent experiments with different parameters ε and δ on the benchmark functions with equality costraints (g03,g05,g11 and g13)

表 7 针对等式约束 Benchmark 函数 g03,g05,g11 和 g13,每个参数δ和ε进行 50 次独立实验的统计结果

Fcn/Optimal	Status	Different parameter values						
		ε=0.1, δ=0.1	ε=0.5, δ=0.5	ε=1, δ=1	ε=5, δ=5	ε=10, δ=10	ε=15, δ=15	ε=20, δ=20
g03/	Times	36	50	50	50	50	50	50
g05/	Times	32	50	50	50	50	50	50
g11/	Times	50	50	50	12	0	0	0
g13/	Times	6	10	11	24	46	45	47

参数δ表示等式约束放松的程度,它使进化过程从等式约束两边满足该约束.等式约束使可行域占整个搜索空间比例非常小,若不放松等式约束,则找到可行域的概率非常小.参数δ越大,表示可以从违反等式约束越远的地方寻找最优解,从而加大了搜索的范围,故有更大概率获得最优解.g03,g05,g13 函数随着δ的增大,50 次实验达到最优解的次数 Times 在不断增大.这证明随着δ的增大,找到最优解的概率增加.参数δ变小,会减小等式约束附近搜索范围,从而减小找到满足等式约束可行域的概率.

4.4.3 参数δ越大越好吗?

表 7 显示,g11 随着δ的增大,最优解却在变差.这个现象与第 4.4.2 节的结论相矛盾.

为了探寻原因,我们分析当参数较大时,函数 g11 一次典型的进化过程.表 8 展示了函数 g11 针对δ=20 和ε=20 时,随着进化代数的增加,种群中可行解的均值 *feasible_mean*、不可行解的均值 *unfeasible_mean*、不可行解距离可行域边界的距离 *unfeasible_distance* 和不可行解的数目 *unfeasible_number* 的变化.

Table 8 A typical evolutionary process with ε=20 and δ=20 on g11

表 8 Benchmark 函数 g11 参数ε=20,δ=20 的一个典型进化过程

gen	*feasible_mean*	*unfeasible_mean*	*unfeasible_distance*	*unfeasible_number*
1	1.7354	NaN	NaN	0
51	5.84E−24	NaN	NaN	0
101	NaN	4.22E−41	0.38048	200
151	NaN	4.70E−33	0.88907	200
201	NaN	4.52E−25	0.98014	200
251	NaN	1.25E−15	0.99644	200
301	NaN	2.45E−06	0.99936	200
351	NaN	0.00070685	0.99918	200
401	NaN	0.15578	0.8442	200
451	1	NaN	NaN	0
501	1	NaN	NaN	0
1001	1	NaN	NaN	0
2001	1	NaN	NaN	0
3001	1	NaN	NaN	0
4001	1	NaN	NaN	0
5001	1	NaN	NaN	0
6001	1	NaN	NaN	0
7001	1	NaN	NaN	0
8001	1	NaN	NaN	0
9001	1	NaN	NaN	0

表 8 存在两个值得关注的现象:第一,当迭代次数 101~401 之间时,种群中不可行解数目 *unfeasible_number*=200.这表示整个种群全部是不可行解,进化过程只在不可行域一侧进化;第二,当迭代次数 201~351 之间时,种群

中不可行解均值 *unfeasible_mean* 始终在 0 附近徘徊,而且种群中不可行解距离可行域边界的距离 *unfeasible_distance* 始终在 1 附近.这表示种群基本已经收敛到不可行域内的局部最优解.由以上分析可知,等式约束的放松程度δ过大,使整个种群只在不可行域一侧进化,而且还没进化到可行域就已经收敛.

4.4.4 参数ε和δ的设置原则

综上所述,得出以下参数设置原则:

原则 1. 对于不存在等式约束的 COPs,ε=1 是恰当的选择,且ε变大不会影响算法的结果.

原则 2. 对于存在等式的 COPs,一般情形下,δ=1 且ε=1 是合适的选择.

原则 3. 对于存在等式约束的 COPs,如果δ=1 不能找到可行解,则δ的设置存在两条经验:第一是设置δ足够大,从而能够保证找到可行域,例如 g13,δ必须比较大时才能保证找到可行解;第二是δ数值不能过大,例如 g11,δ数值过大,会造成进化过程中只从不可行域一侧进化,收敛到局部最优解.

5 结 论

本文基于种群约束允许放松程度ε的概念,改进了 Deb 的 3 条比较准则,结合改进 DE 算法提出了一种新颖的约束处理算法,并成功解决了 13 个 Benchmark 函数中的 11 个.本文的约束处理方法对于 COPs 的解决提供了一种新思路.

仿真实验的结果显示,ε-DE 算法解决 COPs 问题非常有竞争力;50 次实验的标准差显示,该算法具有很强的鲁棒性.

实验结果还显示,ε-DE 算法依然没能完全解决 13 个 Benchmark 函数中的 g02 和 g13.g02 的问题主要是测试算法在可行域内的寻优能力,为了进一步提升算法的效果,必须对 DE 算法进行改进,增强其寻优能力.g13 问题的重点是如何找到合适的可行域,求解此问题时,ε-DE 算法的成功率是 94%.这说明算法的约束处理部分还存在不一致性.如何进一步提升约束处理技术的一致性,还需进一步研究.

References:

[1] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997,11(4):341−359. [doi: 10.1023/A:1008202821328]

[2] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 2000,186(2-4):311−338. [doi: 10.1016/S0045-7825(99)00389-8]

[3] Wang Y, Cai ZX, Zhou YR, Xiao CX. Constrained optimization evolutionary algorithms. Journal of Software, 2009,20(1):11−29 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3363.htm [doi: 10.3724/SP.J.1001.2009.03363]

[4] 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]

[5] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans. on Evolutionary Computation, 2000,4(3):284−294. [doi: 10.1109/4235.873238]

[6] Price KV. Differential evolution vs the functions of the 2nd ICEO. In: Bäck T, Michalewicz Z, Yao X, eds. Proc. of the 1997 IEEE Int'l Conf. on Evolutionary Computation (ICEC'97). Piscataway: IEEE Service Center, 1997. 153−157. [doi: 10.1109%2fICEC.1997.592287]

[7] Storn R. System design by constraint adaptation and differential evolution. IEEE Trans. on Evolutionary Computation, 1999,3(1): 22−34. [doi: 10.1109/4235.752918]

[8] Lampinen J. A constraint handling approach for the differential evolution algorithm. In: Yao X, ed. Proc. of the 2002 Congress on Evolutionary Computation (CEC 2002). Piscataway: IEEE Service Center, 2002. 1468−1473. [doi: 10.1109/CEC.2002.1004459]

[9] Lin YC, Hwang KS, Wang FS. Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization problems. In: Yao X, ed. Proc. of the 2002 Congress on Evolutionary Computation (CEC 2002). Piscataway: IEEE Service Center, 2002. 872−877. [doi: 10.1109/CEC.2002.1007040]

[10] Mezura-Montes E, Velazquez-Reyes J, Coello CAC. Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization. In: Beyer HG, ed. Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2005). New York: ACM Press, 2005. 225−232. [doi: 10.1145/1068009.1068043]

[11] Lin D, Li MQ, Kou JS. A GA-based method for solving constrained OPT imization problems. Journal of Software, 2001,12(4): 628−632 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/12/628.htm

[12] Runarsson TP, Yao X. Search biases in constrained evolutionary optimization. IEEE Trans. on Systems Man and Cybernetics Part C—Applications and Reviews, 2005,35(2):233−243. [doi: 10.1109/TSMCC.2004.841906]

附中文参考文献:

[3] 王勇,蔡自兴,周育人,肖赤心.约束优化进化算法.软件学报,2009,20(1):11−29. http://www.jos.org.cn/1000-9825/3363.htm [doi: 10.3724/SP.J.1001.2009.03363]

[11] 林丹,李敏强,寇纪淞.基于遗传算法求解约束优化问题的一种算法.软件学报,2001,12(4):628−632. http://www.jos.org.cn/1000-9825/12/628.htm

郑建国(1962－),男,福建永定人,博士,教授,博士生导师,主要研究领域为智能信息处理,数据挖掘.

刘荣辉(1972－),男,博士,讲师,主要研究领域为进化计算,智能信息处理.

王翔(1976－),男,博士,讲师,主要研究领域为进化计算,贝叶斯网络.