2025 年 第 25 卷 第 22 期
2025, 25(22): 09417-10
科 学 技 术 与 工 程
Science Technology and Engineering
ISSN 1671—1815
CN 11—4688/T

DOI:10.12404/j.issn.1671-1815.2406831
引用格式:林家馨, 梁昔明, 龙文. 求解约束优化问题的野狗优化算法[J]. 科学技术与工程, 2025, 25(22): 9417-9426.
Lin Jiaxin, Liang Ximing, Long Wen. Dingo optimization algorithm for solving constrained optimization problems[J]. Science Technology and Engineering, 2025, 25(22): 9417-9426.

自动化技术、计算机技术

求解约束优化问题的野狗优化算法

林家馨[1], 梁昔明[1*], 龙文[2]
(1. 北京建筑大学理学院, 北京 102616; 2. 贵州财经大学数学与统计学院, 贵阳 550025)

摘 要 针对约束优化问题的求解,提出了一种结合 ε 约束处理法和纵横交叉策略的野狗优化算法(εCDOA)。算法先在野狗优化算法的所有个体选择捕猎策略更新位置后引入纵横交叉策略,以提高所得算法的全局搜索和局部搜索能力,也有利于算法跳出局部最优;然后按照 ε 约束处理法,将等式约束转化为不等式约束,并用 ε 水平比较方法代替适应度值比较来评判算法中野狗个体的优劣;最后根据个体的约束违约度按自适应的 ε 值把种群分成两个子群,分别以子群的存活策略计算个体的存活率并更新存活率低的个体。对 CEC 2006 中 19 个标准约束优化问题进行数值实验,数值实验结果显示,算法 εCDOA 比结合 ε 约束处理法的野狗优化算法等 4 种对比算法有更好的寻优效果。对于 3 种经典工程设计问题,算法 εCDOA 给出的设计方案明显优于其他对比算法给出的方案。

关键词 野狗优化算法; 纵横交叉; 约束优化问题; ε 约束处理法; 工程设计
中图法分类号 TP301.6; **文献标志码** A

Dingo Optimization Algorithmfor Solving Constrained Optimization Problems

LIN Jia-xin[1], LIANG Xi-ming[1*], LONG Wen[2]
(1. School of Science, Beijing University of Civil Engineering and Architecture, Beijing 102616, China;
2. School of Mathematics and Statistics, Guizhou University of Finance and Economics, Guiyang 550025, China)

[**Abstract**] In order to solve constrained optimization problems, a dingo optimization algorithm with ε constrained method and crisscross strategy (εCDOA) was proposed. The algorithm first introduces the crisscross strategy in dingo optimization algorithm after all individuals select one hunting strategy and update their positions, so as to improve the global and local search capabilities of the obtained algorithm, which also help the algorithm jump out of the local optimum. Then, according to ε constrained method, the equality constraints were transformed into the inequality constraints, and the ε level comparison method was used instead of fitness value comparison to evaluate the qualities of the dingoes. Finally, based on the individuals' constraint violations, the population is divided into two subgroups according to adaptive ε values. The individuals' survival rates were calculated using the survival strategy of each subgroup, and the individuals with low survival rates were updated. The results of numerical experiments on 19 standard constrained optimization problems in CEC 2006 show that algorithm εCDOA has better optimization performance than four comparative algorithms such as dingo optimization algorithm with ε constrained method. For three classical engineering design problems, the design schemes given by algorithm εCDOA are obviously better than those given by other algorithms.

[**Keywords**] DOA (dingo optimization algorithm); crisscross strategy; constrained optimization problem; ε constrained method; engineering design

在工程系统的设计、建造和维护中,工程师需要做出许多技术和管理决策,这些决策的最终目标一般是最小化所需的成本或者最大化期望的利益[1]。优化是寻找给出函数最大值或最小值的条件的过程,于是人们将优化广泛应用于工程问题中,如航空航天工程中的机翼设计、土木工程中的

收稿日期: 2024-09-11; **修订日期**: 2025-05-13
基金项目: 国家自然科学基金(12361106);贵州省自然科学基金重点项目(黔科合基础-ZK[2023]重点 003);中央支持地方科研创新团队项目(PXM2013_014210_000173);北京建筑大学 2021 年校级教育科学研究项目(Y2113)
第一作者: 林家馨(1999—),女,汉族,河南南阳人,硕士研究生。研究方向:最优化方法及其应用。E-mail:839377620@qq.com。
*通信作者: 梁昔明(1967—),男,汉族,湖南汨罗人,博士,教授,硕士研究生导师。研究方向:最优化方法及其应用、复杂系统建模、优化与控制。E-mail:liangximing@bucea.edu.cn。

结构设计和机械工程中的机械设计[2]。为了方便求解,许多工程设计问题可以用数学公式表示为单目标约束优化问题,即通过改变一组设计变量(如零件尺寸、材料特性等)来最小化(或最大化)受性能约束的目标函数,其中,这些约束来自设计要求,例如产品性能或物理尺寸[3]。

实际中的单目标约束优化问题较为复杂,一般具有以下特点:问题具有维数高、多极值等性质;目标函数或约束函数是复杂非线性的;可行域占决策空间的比例极小;决策空间内具有多个不相连的可行域;最优解位于可行域的边界上等[4],因此求解约束优化问题是十分困难的。求解优化问题的方法有很多,可分为传统优化算法和群智能优化算法两大类。传统的优化算法,如最速下降法、共轭梯度法等,需要梯度信息,适用范围有限,于是群智能优化算法得以发展,如灰狼算法[5]、麻雀搜索算法[6]等。此类算法通过模拟自然界中生物种群的习性,只需目标函数值信息进行位置更新就能找到所求优化问题的最优解,适用范围更为广泛。然而群智能优化算法通常用于求解无约束优化问题,为了将其应用于约束优化问题的求解,近些年来许多学者提出了不同的约束处理方法,将这些方法与求解无约束优化问题的群智能算法结合便得到求解约束优化问题的算法。根据处理约束方式的不同,将约束处理方法分成 6 类:罚函数法[7]、可行性法则[8]、随机排序法[9]、ε 约束处理法[10]、多目标优化法[11]和混合法[12]。方林等[13]将罚函数法与改进天鹰优化器融合得到了一种新的优化方法,用此方法求解根据抱杆工程实例建立的轻量化设计模型得到了更佳的设计方案,使抱杆设计的安全性和经济性得到了提高。张美洲等[14]提出了一个基于三空间分割的遗传算法,引入惩罚函数处理约束条件,并将算法应用于环氧乙烷灭菌装载问题这一约束优化问题,具有实际意义。王文川等[15]提出一种改进的 ε 约束处理法,并与改进的差分进化算法结合,在相关测试集上进行数值实验验证了算法的优越性,并将算法应用于水库群防洪调度优化模型。Ji 等[16]提出了一种基于自适应梯度下降的修复方法,利用目标函数信息,使用梯度下降修正不可行的解决方案,并提出了一种改进的 ε 约束多目标差分进化算法,用于解决约束优化问题。Deb 等[17]将多目标优化和惩罚函数方法相结合来处理约束优化问题,在每一次迭代中,先用双目标优化法估计惩罚系数,再利用罚函数法将约束问题转化成无约束优化问题,然后用局部搜索模型求解转化后的无约束优化问题,最终找到原约束优化问题的最优解。

求解约束优化问题的算法性能的好坏取决于两个因素:一是作为搜索引擎的智能算法,二是用来处理约束的约束处理方法[4]。由此可知,对智能算法进行恰当的改进也能使求解约束优化问题的算法获得更好的效果。野狗优化算法(dingo optimization algorithm)是一种新型生物启发算法[18],其模拟了澳大利亚野狗的捕猎策略,即群体攻击、迫害攻击和清扫行为,并引入了存活策略。该算法原理简单、有较强的寻优能力和数值稳定性,但也有群智能优化算法常见的缺点,如容易陷入局部最优、求解精度较低等。

为了改良野狗优化算法并将其应用于约束优化问题,本文研究在野狗优化算法中引入纵横交叉策略并与 ε 约束处理方法结合提出一种新的算法。在野狗优化算法中所有个体选择捕猎策略进行位置更新后加入纵横交叉策略,以增加种群多样性,并提高所得算法(CDOA)跳出局部最优的能力;结合 ε 约束处理方法和所得改进野狗优化算法(CDOA)形成求解约束优化问题的改进野狗算法(εCDOA)。通过 19 个标准约束优化问题的数值实验结果表明,与所有对比算法相比,算法 εCDOA 有更好的求解精度和寻优效率,并且该算法在求解工程设计问题时能找到更好的设计方案。

1 相关基础知识

1.1 单目标约束优化问题数学模型

单目标约束优化问题的数学模型为

$$\min f(x)\quad \text{s.t.}\begin{cases} g_j(x)\leqslant 0, & j=1,2,\cdots,l \\ h_j(x)=0, & j=l+1,l+2,\cdots,m \end{cases} \tag{1}$$

式(1)中:$x=[x_1,x_2,\cdots,x_{\dim}]\in\Omega$,为决策变量;$f(x)$ 为目标函数;$g_j(x)$ 为不等式约束;$h_j(x)$ 为等式约束;l 为不等式约束的个数;$m-l$ 为等式约束的个数。

求解约束优化问题,不仅要考虑目标函数是否达到最优,还要考虑解是否满足约束条件,即要在可行域(即同时满足不等式约束和等式约束的所有解的集合)中找一个可行点 x^*,使目标函数 $f(x)$ 取得最小值,此时称 $[x^*,f(x^*)]$ 为问题的最优解。

1.2 约束违约度

在约束优化中,等式约束一般被转化为不等式约束,于是,式(1)可转化为

$$\min f(x)\quad \text{s.t.}\begin{cases} g_j(x)\leqslant 0, & j=1,2,\cdots,l \\ |h_j(x)|-\delta\leqslant 0, & j=l+1,\cdots,m \end{cases} \tag{2}$$

式(2)中:δ 为等式约束的容忍参数,一般设为 0.000 1。

所以个体 x 在第 j 个约束条件上的约束违反程度[10]表示为

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

那么,个体 x 总的约束违反程度(又称约束违约度)表示为

$$v(x)=\sum_{j=1}^{m}G_j(x) \tag{4}$$

1.3　ε 约束处理法

ε 约束处理法[10]的核心思想是通过设定的 ε 值,将约束违约度小于 ε 的不可行解视为可行解,并用 ε 水平比较方法评价个体的优劣。该方法用分段函数控制 ε,即

$$\varepsilon(t)=\begin{cases}\varepsilon(0)\left(1-\dfrac{t}{T_c}\right)^{c_p}, & 0<t<T_c\\ 0, & t\geqslant T_c\end{cases} \tag{5}$$

式(5)中:$\varepsilon(0)$ 为所有个体的约束违约度按升序排序后得到的第 $\theta(\theta=0.2N_P)$ 个个体的约束违约度,N_P 为种群规模;t 为当前迭代次数;控制代数 $T_c\in[0.1T_{max},0.8T_{max}]$,其中 T_{max} 是最大迭代次数,$c_p\in[2,10]$。

在 ε 水平比较方法中,个体 x 可行性的优先级高于其目标函数值的最优性。假设 $f_1(f_2)$、$v_1(v_2)$ 分别是个体 $x_1(x_2)$ 的目标函数值和约束违约度,对于任意的 $\varepsilon\geqslant 0$,ε 水平比较方法的定义为

$$(f_1,v_1)<_\varepsilon(f_2,v_2)\Leftrightarrow\begin{cases}f_1<f_2, & v_1,v_2<\varepsilon\\ f_1<f_2, & v_1=v_2\\ v_1<v_2, & \text{其他}\end{cases} \tag{6}$$

$$(f_1,v_1)\leqslant_\varepsilon(f_2,v_2)\Leftrightarrow\begin{cases}f_1\leqslant f_2, & v_1,v_2\leqslant\varepsilon\\ f_1\leqslant f_2, & v_1=v_2\\ v_1\leqslant v_2, & \text{其他}\end{cases} \tag{7}$$

式中:$(f_1,v_1)<_\varepsilon(f_2,v_2)$ 表示 (f_1,v_1) 优于 (f_2,v_2),即对应的个体 x_1 优于个体 x_2;$(f_1,v_1)\leqslant_\varepsilon(f_2,v_2)$ 表示 (f_1,v_1) 优于或等于 (f_2,v_2),即对应的个体 x_1 优于个体 x_2 或与之相当。

1.4　野狗优化算法

1.4.1　3 种捕猎策略

在野狗优化算法的每次迭代中,每只野狗需选择一种捕猎策略进行位置更新,得到的新位置与当前位置进行比较,保留适应度值更小的位置。在 3 种捕猎策略中,群体攻击指野狗在遇到大猎物时会成群结队捕猎,迫害攻击指野狗独自捕获小猎物,清扫行为指野狗的食腐行为[18],群体攻击、迫害攻击和清扫行为的公式为

$$\vec{x}'_i(t)=\beta_1\sum_{k=1}^{n_a}\frac{[\vec{\varphi}_k(t)-\vec{x}_i(t)]}{n_a}-\vec{x}_*(t) \tag{8}$$

$$\vec{x}'_i(t)=\vec{x}_*(t)+\beta_1 e^{\beta_2}[\vec{x}_{r_1}(t)-\vec{x}_i(t)] \tag{9}$$

$$\vec{x}'_i(t)=\frac{1}{2}[e^{\beta_2}\vec{x}_{r_1}(t)-(-1)^{\sigma}\vec{x}_i(t)] \tag{10}$$

式中:t 为当前迭代次数;$\vec{x}_i(t)$ 为第 i 只野狗当前的位置;$\vec{x}'_i(t)$ 为第 i 只野狗选择该策略后生成的新位置;$\vec{x}_*(t)$ 为第 t 次迭代中发现的最佳野狗位置;n_a 为在区间 $[2,N/2]$ 上生成的随机整数;β_1 为在区间 $[-2,2]$ 上均匀生成的随机数,$\vec{\varphi}_k(t)$ 为将要攻击的野狗位置;β_2 为在区间 $[-1,1]$ 上均匀生成的随机数;r_1 为在区间 $[1,N]$ 上生成的随机整数,$r_1\neq i$;$\vec{x}_{r_1}(t)$ 为随机选择的第 r_1 只野狗的位置;σ 为随机生成的二进制数,$\sigma\in\{0,1\}$,在 $[0,1]$ 内生成一个随机数 rand,若 rand$\leqslant$0.5,则 $\sigma=0$,否则 $\sigma=1$。

1.4.2　存活策略

在野狗优化算法中,第 i 只野狗的存活率值为

$$\text{survival}(i)=\frac{\text{fitness}_{max}-\text{fitness}(i)}{\text{fitness}_{max}-\text{fitness}_{min}} \tag{11}$$

式(11)中:fitness_{max} 和 fitness_{min} 分别为当前种群中最差和最佳的个体适应度值;$\text{fitness}(i)$ 为第 i 只野狗的适应度值。

当第 i 只野狗的存活率$\leqslant$0.3 时,用式(12)产生的新位置直接代替其当前位置,即

$$\vec{x}''_i(t)=\vec{x}'_*(t)+\frac{1}{2}[\vec{x}'_{r_1}(t)-(-1)^{\sigma}\vec{x}'_{r_2}(t)] \tag{12}$$

式(12)中:r_1 和 r_2 为在区间 $[1,N]$ 上生成的随机数,$r_1\neq r_2$;$\vec{x}'_{r_1}(t)$ 和 $\vec{x}'_{r_2}(t)$ 为随机选择的第 r_1、r_2 只野狗的位置;$\vec{x}'_*(t)$ 为在结束选择捕猎策略后的野狗种群中发现的最佳野狗的位置;σ 为随机生成的二进制数,$\sigma\in\{0,1\}$,即第 i 只野狗在第 $t+1$ 次迭代的位置。

$$\vec{x}_i(t+1)=\begin{cases}\vec{x}''_i(t), & \text{survival}(i)\leqslant 0.3\\ \vec{x}'_i(t), & \text{survival}(i)>0.3\end{cases} \tag{13}$$

2　融合纵横交叉策略的野狗优化算法

为了克服野狗优化算法(DOA)易陷入局部最

优的不足,引入纵横交叉策略得到一种新的改进野狗优化算法(CDOA)。

2.1 横向交叉

纵横交叉策略[19]由横向交叉和纵向交叉两种方式组成。横向交叉是两个不同个体在所有维度上进行的算术交叉,可以减少个体的搜索盲区,增强算法的全局搜索能力。

交叉之前先对所有个体进行随机两两不重复配对,假设个体 $X(i)$ 与 $X(j)$ 相配对作为父代在第 d 维上执行横向交叉策略,那么它们的子代 $\mathrm{MS}_{\mathrm{hc}}(i)$ 和 $\mathrm{MS}_{\mathrm{hc}}(j)$ 的公式为

$$\mathrm{MS}_{\mathrm{hc}}(i,d) = r_1X(i,d) + (1-r_1)X(j,d) + c_1[X(i,d) - X(j,d)] \tag{14}$$

$$\mathrm{MS}_{\mathrm{hc}}(j,d) = r_2X(j,d) + (1-r_2)X(i,d) + c_2[X(j,d) - X(i,d)] \tag{15}$$

式中:$d = 1,2,\cdots,\dim$,其中 dim 为个体的维数;r_1 和 r_2 为[0,1]上均匀分布的随机数;c_1 和 c_2 为[-1,1]上均匀分布的随机数;$X(i,d)$、$X(j,d)$ 分别为父代 $X(i)$ 和 $X(j)$ 的第 d 维;$\mathrm{MS}_{\mathrm{hc}}(i,d)$、$\mathrm{MS}_{\mathrm{hc}}(j,d)$ 分别为子代 $\mathrm{MS}_{\mathrm{hc}}(i)$ 和 $\mathrm{MS}_{\mathrm{hc}}(j)$ 的第 d 维,生成的子代 $\mathrm{MS}_{\mathrm{hc}}(i)$ 和 $\mathrm{MS}_{\mathrm{hc}}(j)$ 分别与其父代进行比较,保留适应度值更小的个体。

2.2 纵向交叉

纵向交叉是所有个体在不同维之间进行的算术交叉,可以帮助算法跳出局部最优。每个个体进行一次纵向交叉,只对其中某一维进行更新,其他维保持不变。

假设 d_1、d_2 为个体 $X(i)$ 的两个不同维度,那么执行纵向交叉后产生的子代 $\mathrm{MS}_{\mathrm{vc}}(i)$ 的第 d_1 维如式(16)所示,其他维与父代 $X(i)$ 保持不变。

$$\mathrm{MS}_{\mathrm{vc}}(i,d_1) = rX(i,d_1) + (1-r)X(i,d_2) \tag{16}$$

式(16)中:r 为[0,1]上均匀分布的随机数,生成的子代 $\mathrm{MS}_{\mathrm{vc}}(i)$ 与其父代进行比较,保留适应度值更小的个体。

在整个野狗种群选择捕猎策略更新位置后加入纵横交叉策略,所有个体先执行横向交叉,再执行纵向交叉,形成的融合纵横交叉策略的野狗优化算法(CDOA)的流程图如图 1 所示。

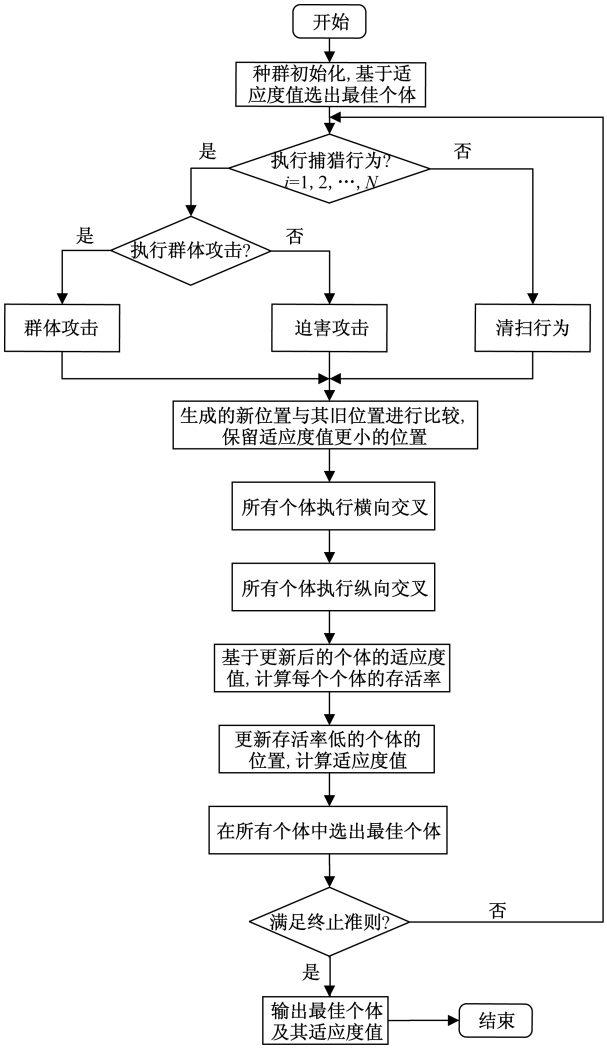

图 1 算法 CDOA 流程图

Fig. 1 The flowchart of algorithm CDOA

3 求解约束优化问题的野狗优化算法

为求解数学模型如式(1)的约束优化问题,需对求解无约束优化问题的算法 CDOA 做以下修正,并与 ε 约束处理法结合,从而得到结合 ε 约束处理法和纵横交叉策略的野狗优化算法 εCDOA。

(1)用 ε 水平比较方法代替算法 CDOA 中的适应度值比较来评价个体优劣。

(2)对算法 CDOA 中的存活策略进行修正得到结合 ε 约束处理法的存活策略。所有个体执行纵横交叉后,其位置表示为 $\vec{x}''_1,\vec{x}''_2,\cdots,\vec{x}''_{N_p}$,根据个体约束违约度是否 $\leqslant\varepsilon$,将野狗种群分成两个子群 Ω_1 和 Ω_2,其中属于 Ω_1 的野狗个体的约束违约度 $\leqslant\varepsilon$,属于 Ω_2 的野狗个体的约束违约度 $>\varepsilon$。

设 p 是子群 Ω_1 的种群规模,记 Ω_1 中个体为 $\vec{x}''_{11}(t),\vec{x}''_{12}(t),\cdots,\vec{x}''_{1p}(t)$。对于 Ω_1 中的第 k 只野狗,其存活率值为

$$\mathrm{survival}(k) = \frac{\mathrm{fitness}_{\max} - \mathrm{fitness}(k)}{\mathrm{fitness}_{\max} - \mathrm{fitness}_{\min}} \tag{17}$$

式(17)中:$\mathrm{fitness}_{\max}$ 和 $\mathrm{fitness}_{\min}$ 分别为当前子群 Ω_1 中最差和最佳个体的适应度值;$\mathrm{fitness}(k)$ 为子群 Ω_1 中第 k 只野狗的适应度值。

当子群 Ω_1 中第 k 只野狗的存活率 $\leqslant$0.3 时,用

式(18)生成的新位置直接替换该野狗的当前位置,即

$$\vec{x}'''_{1k}(t) = \vec{x}''_{1b}(t) + \frac{1}{2}[\vec{x}''_{1\alpha_1}(t) - (-1)^{\sigma}\vec{x}''_{1\alpha_2}(t)] \tag{18}$$

式(18)中:$\vec{x}'''_{1k}(t)$ 为子群 Ω_1 中第 k 只野狗的新位置;α_1 和 α_2 为在 1 和 p 中随机生成的整数,且 $\alpha_1 \neq \alpha_2$;$\vec{x}''_{1\alpha_1}(t)$ 和 $\vec{x}''_{1\alpha_2}(t)$ 为子群 Ω_1 中第 α_1 和 α_2 只野狗的位置;$\vec{x}''_{1b}(t)$ 为子群 Ω_1 中最佳野狗位置;σ 为算法随机生成的二进制数,$\sigma \in \{0,1\}$。

设 q 是子群 Ω_2 的种群规模,记 Ω_2 中个体为 $\vec{x}''_{21}(t),\vec{x}''_{22}(t),\cdots,\vec{x}''_{2q}(t)$。对于子群 Ω_2 中的第 k 只野狗,其存活率值为

$$\text{survival}(k) = \frac{\text{violation}_{\max} - \text{violation}(k)}{\text{violation}_{\max} - \text{violation}_{\min}} \tag{19}$$

式(19)中:$\text{violation}_{\max}$ 和 $\text{violation}_{\min}$ 分别为当前子群 Ω_2 中个体的最大和最小约束违约度;$\text{violation}(k)$ 为 Ω_2 中第 k 只野狗的约束违约度。

当子群 Ω_2 中第 k 只野狗的存活率≤0.3 时,用式(20)生成的新位置直接替换该野狗的当前位置,即

$$\vec{x}'''_{2k}(t) = \vec{x}''_{2b}(t) + \frac{1}{2}[\vec{x}''_{2\alpha_3}(t) - (-1)^{\sigma}\vec{x}''_{2\alpha_4}(t)] \tag{20}$$

式(20)中:$\vec{x}'''_{2k}(t)$ 为子群 Ω_2 中第 k 只野狗的新位置;α_3 和 α_4 为在 1 和 q 中随机生成的整数,且 $\alpha_3 \neq \alpha_4$;$\vec{x}''_{2\alpha_3}(t)$ 和 $\vec{x}''_{2\alpha_4}(t)$ 为子群 Ω_2 中第 α_3 和 α_4 只野狗的位置;$\vec{x}''_{2b}(t)$ 为子群 Ω_2 中最佳野狗位置;σ 为算法随机生成的二进制数。

结合 ε 约束处理法和纵横交叉策略的野狗优化算法(εCDOA)的伪代码如表 1 所示。

4　数值实验

为了验证所得算法 εCDOA 的有效性,对文献[20]中 19 个标准约束优化问题 $g01 \sim g19$ 进行了测试,其中问题 $g03$、$g11$、$g13 \sim g15$ 和 $g17$ 的约束条件均为等式约束,问题 $g05$ 的约束条件既有等式约束也有不等式约束,其他问题的约束条件均为不等式约束。应用算法 εCDOA 求解 19 个问题的测试结果与结合 ε 约束处理法的基本野狗优化算法(εDOA)及其他相关算法的结果进行了对比分析。

4.1　算法 εDOA 与算法 εCDOA 的数值比较

实验中,设置两种算法的种群规模 N_P 均为 50,最大迭代次数 $T_{\max}$ 均为 1 000,概率 P 和 Q 分别为 0.5 和 0.7,ε 约束处理法中的 T_c 和 c_p 分别设为 $0.5T_{\max}$ 和 5。对每个标准约束优化问题,两种算法均独立运行 30 次,取所得目标函数值的最好值、最差值、平均值、标准差进行对比,数值实验结果如表 2所示,表中所得目标函数值对应的解均为相应问题的可行解。

由表 2 可知,对于 19 个标准约束优化问题,算法 εCDOA 的总体数值表现优于算法 εDOA,其中求解 13 个问题($g01 \sim g03$、$g05$、$g07$、$g09$、$g10$、$g13$、$g14$、$g16 \sim g19$)所得目标函数值的最好值、最差值、平均值、标准差均优于算法 εDOA 所得的相应值。对于问题 $g01$、$g04$、$g06$、$g11$,算法 εCDOA 能求得问题的理论最优目标函数值,寻优能力更强,且所得标准差均小于 10^{-7},远小于算法 εDOA 的标准差,说明算法 εCDOA 在求解这些问题时更精确、更

表 1　算法 εCDOA 伪代码

Table 1　The pseudocode of algorithm εCDOA

```
1. 设置算法参数,如种群规模 N_P、最大迭代次数 T_max、选择捕猎行为或清扫行为的概率 P、选择群体攻击或迫害攻击的概率 Q、ε 约束处理法中的 T_c 和 c_p;
2. 种群初始化;
3. 计算所有个体的适应度值和约束违约度,用 ε 水平比较方法确定最佳个体和 ε(0);
4. t = 1;
5. while t < T_max
6.    根据式(5)更新 ε 的值;
7.    for i = 1:N_P
8.        if random < P
9.            if random < Q
10.               按群体攻击策略生成一个新位置,用 ε 水平比较方法与当前位置比较,保留更优的位置;
11.           else
12.               按迫害攻击策略生成一个新位置,用 ε 水平比较方法与当前位置比较,保留更优的位置;
13.           end if
14.       else
15.       按清扫行为策略生成一个新位置,用 ε 水平比较方法与当前位置比较,保留更优的位置;
16.       end if
17.   end for
18.   所有个体执行横向交叉,用 ε 水平比较方法比较新位置与当前位置,保留更优位置;
19.   所有个体执行纵向交叉,用 ε 水平比较方法比较新位置与当前位置,保留更优位置;
20.   根据约束违约度是否小于等于 ε,将种群分成两部分,分别计算两个子群中每个个体的存活率,并更新存活率 ≤ 0.3 的个体;
21.   确定当前种群中的最佳个体;
22.   t = t + 1;
23. end while
24. 输出最佳个体及其适应度值。
```

表 2 算法 εDOA 与 εCDOA 所得的目标函数值

Table 2 The objective function values obtained by algorithm εDOA and εCDOA

问题	理论最优值	算法	最好值	最差值	平均值	标准差
g01	-15	ε DOA	-14.004 1	-3	-6.267 2	2.743 2
		ε CDOA	**-15**	**-15**	**-15**	**0**
g02	-0.803 6	ε DOA	-0.571 8	-0.370 0	-0.447 4	0.052 4
		ε CDOA	**-0.794 6**	**-0.705 7**	**-0.760 4**	**0.023 4**
g03	-1.000 5	ε DOA	-0.933 8	-3.79×10^{-5}	-0.511 8	0.340 3
		ε CDOA	**-1.000 3**	**-5.08×10^{-18}**	**-0.897 3**	**0.261 9**
g04	-30 665.538 7	ε DOA	**-30 665.538 7**	-30 183.519 8	-30 629.685 7	121.735 1
		ε CDOA	**-30 665.538 7**	**-30 665.538 7**	**-30 665.538 7**	**1.11×10^{-11}**
g05	5 126.496 7	ε DOA	5 126.727 5	6 095.471 4	5 502.670 6	375.782 8
		ε CDOA	**5 126.496 7**	**5 127.545 0**	**5 126.625 5**	**0.259 8**
g06	-6 961.813 9	ε DOA	**-6 961.813 9**	-6 961.794 1	-6 961.810 6	0.006 8
		ε CDOA	**-6 961.813 9**	**-6 961.813 9**	**-6 961.813 9**	**2.37×10^{-8}**
g07	24.306 2	ε DOA	29.805 1	1 299.486 1	276.569 9	313.666 2
		ε CDOA	**24.319 4**	**25.265 1**	**24.659 0**	**0.232 5**
g08	-0.095 8	ε DOA	**-0.095 8**	**-0.095 8**	**-0.095 8**	**7.29×10^{-18}**
		ε CDOA	**-0.095 8**	**-0.095 8**	**-0.095 8**	**1.61×10^{-17}**
g09	680.630 1	ε DOA	681.300 5	749.253 7	695.897 9	18.312 5
		ε CDOA	**680.630 5**	**680.661 3**	**680.634 3**	**0.005 7**
g10	7 049.248 0	ε DOA	8 324.764 8	15 351.590 0	10 494.268 1	1 852.004 2
		ε CDOA	**7 109.070 2**	**8 803.482 8**	**7 536.556 6**	**436.906 5**
g11	0.749 9	ε DOA	**0.749 9**	0.857 0	0.755 4	0.022 0
		ε CDOA	**0.749 9**	**0.749 9**	**0.749 9**	**1.13×10^{-16}**
g12	-1	ε DOA	**-1**	**-1**	**-1**	**0**
		ε CDOA	**-1**	**-1**	**-1**	**0**
g13	0.053 9	ε DOA	0.069 5	1.545 5	0.651 3	0.313 6
		ε CDOA	**0.053 9**	**0.438 9**	**0.066 8**	**0.070 3**
g14	-47.764 9	ε DOA	-47.456 7	-39.403 4	-44.249 8	2.358 7
		ε CDOA	**-47.710 1**	**-46.805 2**	**-47.409 7**	**0.263 3**
g15	961.715 0	ε DOA	**961.715 0**	971.812 8	964.294 6	3.117 7
		ε CDOA	**961.715 0**	**961.715 5**	**961.715 0**	**8.70×10^{-5}**
g16	-1.905 2	ε DOA	-1.904 4	-1.700 2	-1.891 2	0.038 4
		ε CDOA	**-1.905 2**	**-1.905 1**	**-1.905 2**	**9.39×10^{-6}**
g17	8 853.539 7	ε DOA	8 975.693 6	9 184.169 8	9 081.076 0	108.806 2
		ε CDOA	**8 866.808 3**	**9 018.070 2**	**8 961.019 2**	**57.446 3**
g18	-0.866 0	ε DOA	-0.844 8	-0.544 2	-0.725 9	0.076 1
		ε CDOA	**-0.866 0**	**-0.864 2**	**-0.865 5**	**0.000 5**
g19	32.655 6	ε DOA	41.299 4	150.423 8	78.525 3	21.740 0
		ε CDOA	**32.787 8**	**40.467 9**	**34.875 7**	**1.664 4**

注:加粗数值表示更优结果。

稳定。对于问题 g05、g07、g10、g19,算法 εCDOA 在独立运行 30 次中得到的目标函数值平均值和标准差明显小于算法 εDOA 的相应值,与算法 εDOA 相比表现出明显的优势。对于问题 g15、g16,算法 εCDOA 虽然未能在 30 次独立求解中均求得其理论最优目标函数值,但所得目标函数值平均值的精度可达到 10^{-4},高于算法 εDOA 的相应精度,且其标准差可达到 10^{-5}和 10^{-6},说明算法 εCDOA 有更好的数值稳定性。对于问题 g08 和 g12,算法 εCDOA 和算法 εDOA 在 30 次独立求解中均能求得理论最优目标函数值,算法 εCDOA 和算法 εDOA 有相同的数值精度。

综上所述,对文献[20]中 19 个标准约束优化问题,与算法 εDOA 相比,算法 εCDOA 有更好的数值精度和数值稳定性。

4.2 算法 εCDOA 与相关算法的数值比较

对于 CEC 2006 中的标准约束优化问题 g01 ~ g13,分别用算法 εCDOA、粒子群算法 PSO[21]、混沌灰狼算法 CGWO[22]、基于 E-BRM 遗传算法[23]求解,取所得目标函数值的平均值和标准差进行比较,数值实验结果如表 3 所示。

由表 3 可知,对于标准约束优化问题 g01 ~ g13,算法 εCDOA 能找到问题 g01、g04、g06、g08、

表 3　算法 εCDOA 与对比算法所得的目标函数值

Table 3　The objective function values obtained by algorithm εCDOA and comparative algorithms

问题	理论最优值	参数	PSO	CGWO	E-BRM	ε CDOA
$g01$	−15	平均值	−14.715 1	−14.990 5	−14.998 85	**−15**
		标准差	7.40×10^{-1}	6.30×10^{-3}	6.00×10^{-4}	0
$g02$	−0.803 6	平均值	−0.740 6	**−0.804 1**	−0.801 5	−0.760 4
		标准差	4.20×10^{-2}	5.70×10^{-3}	3.83×10^{-3}	0.023 4
$g03$	−1.000 5	平均值	**−1.003 4**	−0.987 8	−0.976 3	−0.897 3
		标准差	1.70×10^{-2}	2.09×10^{-1}	4.40×10^{-2}	0.261 9
$g04$	−30 665.538 7	平均值	**−30 665.539**	−31 492.2	−30 953.533	**−30 665.538 7**
		标准差	0	1.37×10^{2}	6.27	1.11×10^{-11}
$g05$	5 126.496 7	平均值	5 202.362 7	5 432.4	5 304.65	**5 126.625 5**
		标准差	1.10×10^{2}	2.12×10^{2}	1.88×10^{2}	0.259 8
$g06$	−6 961.813 9	平均值	**−6 961.814**	−6 591.482	−6 991.75	**−6 961.813 9**
		标准差	0	5.64×10^{1}	5.10×10^{-2}	2.37×10^{-8}
$g07$	24.306 2	平均值	24.989	42.121	26.463 1	**24.65 9**
		标准差	0.551	8.26	1.09	0.232 5
$g08$	−0.095 8	平均值	**−0.095 8**	−321.042	**−0.095 8**	**−0.095 8**
		标准差	0	6.25	0	1.61×10^{-17}
$g09$	680.630 1	平均值	680.653	679.58	681.054 1	**680.634 3**
		标准差	1.80×10^{-2}	1.76×10^{1}	0.232	0.005 7
$g10$	7 049.248 0	平均值	7 173.266 1	**7 043.39**	10412.32	7 536.556 6
		标准差	8.40×10^{2}	3.61×10^{1}	3.11×10^{2}	436.906 5
$g11$	0.749 9	平均值	0.749	0.719	0.880 5	**0.749 9**
		标准差	3.90×10^{-7}	1.28×10^{-2}	7.95×10^{-2}	1.13×10^{-16}
$g12$	−1	平均值	**−1**	−21.154	**−1**	**−1**
		标准差	0	5.65	0	0
$g13$	0.053 9	平均值	0.552 753	0.379 1	0.891 9	**0.066 8**
		标准差	4.20×10^{-1}	6.80×10^{-2}	3.50×10^{-1}	0.070 3

注:加粗数值表示所有算法中的最好结果。

$g11$ 和 $g12$ 的理论最优值。算法 εCDOA 在求解 $g01$、$g05$、$g07$、$g09$、$g11$、$g13$ 这 6 个问题上所得目标函数值的平均值和标准差均优于算法 PSO、算法 CGWO 和算法 E-BRM 的平均值和标准差,说明算法 εCDOA 有更好的寻优能力和数值稳定性。对于问题 $g04$、$g06$ 和 $g08$,算法 εCDOA 得到的目标函数值平均值均可达到理论最优目标函数值,优于或等于其他 3 种对比算法给出的平均值,即算法 εCDOA 在这些问题上的寻优表现优于 3 种对比算法或与 3 种对比算法相当,其中算法 PSO 也能找到问题的理论最优值,但算法 εCDOA 的标准差略大于算法 PSO 的标准差,说明在求解这些问题时算法 εCDOA 的数值稳定性略逊于算法 PSO。对于问题 $g12$,算法 εCDOA 和算法 PSO、算法 E-BRM 有相同的求解结果,所得目标函数值均为相应的理论最优值,且标准差为 0。对于问题 $g02$ 和 $g10$,算法 εCDOA 求得的目标函数值平均值优于其中一种对比算法而较差于另两种对比算法。对于问题 $g03$,εCDOA 算法的寻优表现略差于 3 种对比算法。

综上所述,对 13 个标准约束优化问题中的多数问题,相比于 3 种对比算法,算法 εCDOA 的寻优表现更好。

5　经典工程设计问题

5.1　压力容器设计问题

压力容器是由厚圆柱壳和有厚度的半球体状封头组成,如图 2 所示。该设计问题的目标是使制造容器的总成本最小化[18]。问题有 4 个变量:壳体厚度 T_s、封头厚度 T_h、内半径 R 和不含封头的圆柱部分的长度 L。压力容器设计问题的数学模型为

$$\begin{cases}\vec{x}=[x_1x_2x_3x_4]=[T_sT_hRL]\\ \min f(\vec{x})=0.622\,4x_1x_3x_4+1.778\,1x_2x_3^2+\\ \qquad 3.166\,1x_1^2x_4+19.84x_1^2x_3\\ \text{s.t. } g_1(\vec{x})=-x_2+0.019\,3x_3\leqslant 0\\ \qquad g_2(\vec{x})=-x_1+0.009\,54x_3\leqslant 0\\ \qquad g_3(\vec{x})=-\pi x_3^2x_4-\dfrac{4}{3}\pi x_3^3+\\ \qquad\qquad 1\,296\,000\leqslant 0\\ \qquad g_4(\vec{x})=x_4-240\leqslant 0\end{cases}\tag{21}$$

式(21)中:$0\leqslant x_1,x_2\leqslant 99$;$10\leqslant x_3,x_4\leqslant 200$。

这是一个只有不等式约束的优化问题,可直接运用算法 εCDOA 进行求解。算法 εCDOA 和 10 种对比算法求解问题(21)的结果如表4所示,表4中

表 4 压力容器设计问题的求解结果

Table 4 Numerical results of pressure vessel design problem

算法	T_s	T_h	R	L	f
ε CDOA	0.778 199	0.384 664	40.321 209	199.977 862	5 885.385 5
DOA	0.812 500	0.437 500	42.098 45	176.636 6	6 059.714 3
ACO	0.812 500	0.437 500	42.103 624	176.572 656	6 059.088 8(不可行)
DE	0.812 500	0.437 500	42.098 411	176.637 690	6 059.734 0
WOA	0.812 500	0.437 500	42.098 269 9	176.638 998	6 059.741 0
ES	0.812 500	0.437 500	42.098 087	176.640 518	6 059.745 6
GA	0.812 500	0.437 500	42.097 398	176.654 050	6 059.946 3
PSO	0.812 500	0.437 500	42.091 266	176.746 500	6 061.077 7
GA	0.812 500	0.434 500	40.323 900	200.000 000	6 288.744 5
Improved HS	1.125 000	0.625 000	58.290 150	43.692 680 0	7 197.730
Lagrangian multiplier	1.125 000	0.625 000	58.291 000	43.690 000 0	7 198.042 8

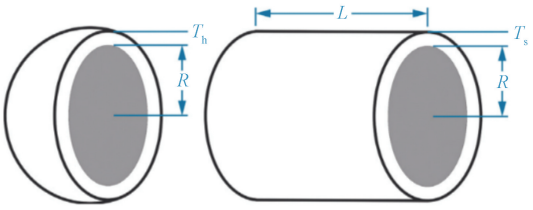

图 2 压力容器设计问题

Fig. 2 Pressure vessel design problem

对比算法的数据均来自文献[18]。从表 4 可以看出,算法 εCDOA 的求解结果明显优于算法 DOA 等所有对比算法的结果,因而它能为压力容器设计问题提供更好的设计方案,使制造成本进一步降低。

5.2 焊接梁设计问题

焊接梁组成如图 3 所示,其设计目标是使焊接梁的制造成本最小化[24]。设计变量有 4 个:焊缝厚度 h、梁条长度 l、梁条高度 t 和梁条厚度 b,约束分别为切应力 τ、弯曲应力 θ、梁条弯曲载荷 P_c 和末端误差 δ。焊接梁设计问题的数学模型为

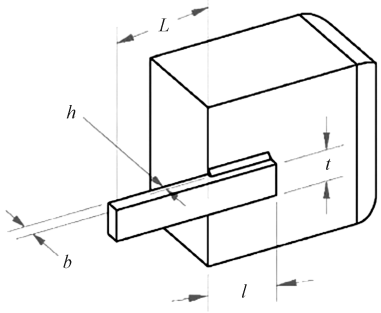

图 3 焊接梁设计问题

Fig. 3 Welded beam design problem

$$\begin{cases} \vec{x} = [x_1x_2x_3x_4] = [hltb] \\ \min f(\vec{x}) = 1.104\,71x_1^2x_2 + 0.048\,11x_3x_4(14.0 + x_2) \\ \text{s.t. } g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \leqslant 0 \\ \quad g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \leqslant 0 \\ \quad g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \leqslant 0 \\ \quad g_4(\vec{x}) = x_1 - x_4 \leqslant 0 \\ \quad g_5(\vec{x}) = P - P_c(\vec{x}) \leqslant 0 \\ \quad g_6(\vec{x}) = 0.125 - x_1 \leqslant 0 \\ \quad g_7(\vec{x}) = 1.104\,71x_1^2x_2 + 0.048\,11x_3x_4 \times (14.0 + x_2) - 5.0 \leqslant 0 \end{cases} \tag{22}$$

式(22)中:

$$\begin{cases} \tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\dfrac{x_2}{2R} + (\tau'')^2} \\ \tau' = \dfrac{P}{\sqrt{2}x_1x_2} \\ \tau'' = \dfrac{MR}{J},\ M = P\left(L + \dfrac{x_2}{2}\right) \\ R = \sqrt{\dfrac{x_2^2}{4} + \left(\dfrac{x_1 + x_3}{2}\right)^2} \\ J = 2\left\{\sqrt{2}x_1x_2\left[\dfrac{x_2^2}{4} + \left(\dfrac{x_1 + x_3}{2}\right)^2\right]\right\} \\ \sigma(\vec{x}) = \dfrac{6PL}{x_4x_3^2} \\ \delta(\vec{x}) = \dfrac{6PL^3}{Ex_3^2x_4} \\ P_c(\vec{x}) = \dfrac{4.013E\sqrt{\dfrac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \dfrac{x_3}{2L}\sqrt{\dfrac{E}{4G}}\right) \end{cases} \tag{23}$$

式中:$P = 6\,000$;$L = 14$;$\delta_{\max} = 0.25$;$E = 30 \times 10^6$;$G = 12 \times 10^6$;$\tau_{\max} = 13\,600$;$\delta_{\max} = 30\,000$;$0.1 \leqslant x_1, x_4 \leqslant 2$;$0.1 \leqslant x_2, x_3 \leqslant 10$。

表 5　焊接梁设计问题的求解结果

Table 5　Numerical results of welded beam design problem

算法	h	l	t	b	f
ε CDOA	0.205 730	3.234 919	9.036 624	0.205 730	1.692 768
ε DOA	0.196 571	3.434 454	9.037 416	0.205 726	1.706 074
WOA	0.205 396	3.484 293	9.037 426	0.206 276	1.730 499
GSA	0.182 129	3.856 979	10	0.202 376	1.879 952
CBO	0.205 722	3.470 41	9.037 276	0.205 735	1.724 663
RO	0.203 687	3.528 467	9.004 233	0.207 241	1.735 344
Improved HS	0.205 73	3.470 49	9.036 62	0.205 7	1.724 8
GA	0.248 9	6.173	8.178 9	0.253 3	2.433 1
HS	0.244 2	6.223 1	8.291 5	0.244 3	2.380 7

这是一个只有不等式约束的约束优化问题,算法 εCDOA 和 8 种对比算法对问题(23)的求解结果如表 5 所示,对比算法的数据均来自文献[24]。

从表 5 可以看出,算法 εCDOA 求解焊接梁设计问题时所得目标函数值优于算法 εDOA 和算法 WOA 等所有对比算法的相应结果,这说明算法 εCDOA能为焊接梁设计问题提供更优的设计方案。

5.3　减速器设计问题

减速器设计问题(图 4)是一个标准工程设计优化问题,涉及 7 个设计变量:端面宽宽度 b、齿模数 m、小齿轮的齿数 z、轴承之间轴 1 的长度 l_1、轴承之间轴 2 的长度 l_2、轴 1 直径 d_1、轴 2 直径 d_2。该问题的目标是使减速器的总重量最小,约束包括对齿轮齿的弯曲应力、表面应力、轴 1 和轴 2 因传递力引起的横向挠度以及轴 1 和轴 2 的应力的限制[25]。减速器设计问题的数学模型为

$$
\begin{cases}
\vec{x} = [x_1x_2x_3x_4x_5x_6x_7] = [bmzl_1l_2d_1d_2] \\
\min f(\vec{x}) = 0.785\,4x_1x_2^2(3.333x_3^2 + 14.934x_3 - 43.093\,4) - 1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.785\,4(x_4x_6^2 + x_5x_7^2) \\
\text{s.t. } g_1(\vec{x}) = 27x_1^{-1}x_2^{-2}x_3^{-1} \leqslant 1 \\
g_2(\vec{x}) = 397.5x_1^{-1}x_2^{-2}x_3^{-2} \leqslant 1 \\
g_3(\vec{x}) = 1.93x_2^{-1}x_3^{-1}x_4^3x_6^{-4} \leqslant 1 \\
g_4(\vec{x}) = 1.93x_2^{-1}x_3^{-1}x_5^3x_7^{-4} \leqslant 1 \\
g_5(\vec{x}) = \dfrac{\left[\left(\dfrac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right]^{0.5}}{0.1x_6^3} \leqslant 1\,100 \\
g_6(\vec{x}) = \dfrac{\left[\left(\dfrac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6\right]^{0.5}}{0.1x_7^3} \leqslant 850 \\
g_7(\vec{x}) = x_2x_3 \leqslant 40 \\
g_8(\vec{x}) = \dfrac{x_1}{x_2} \geqslant 5 \\
g_9(\vec{x}) = \dfrac{x_1}{x_2} \leqslant 12 \\
g_{10}(\vec{x}) = (1.5x_6 + 1.9)x_4^{-1} \leqslant 1 \\
g_{11}(\vec{x}) = (1.1x_7 + 1.9)x_5^{-1} \leqslant 1
\end{cases}
\tag{24}
$$

式(24)中:$2.6 \leqslant x_1 \leqslant 3.6$;$0.7 \leqslant x_2 \leqslant 0.8$;$17 \leqslant x_3 \leqslant 28$;$7.3 \leqslant x_4 \leqslant 8.3$;$7.3 \leqslant x_5 \leqslant 8.3$;$2.9x_6 \leqslant 3.9$;$5.0 \leqslant x_7 \leqslant 5.5$。

该问题的理论最优解是 $x^* = [3.5\quad 0.7\quad 17.0\quad 7.3\quad 7.3\quad 3.35\quad 5.29]$,其目标函数值是 2 985.22。

用算法 εCDOA 求解此问题所得的结果与 6 种对比算法的求解结果如表 6 所示,表 6 中的对比算法的数据均来自文献[25]。

从表 6 可以看出,与 6 种对比算法相比,算法 εCDOA 求解减速器设计问题时所得目标函数值平均值更接近理论最优目标函数值且标准差更小,数值稳定性更优,所以算法 εCDOA 能够提供更优的设计方案使得减速器总重量更小。

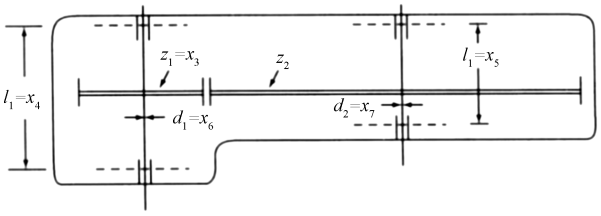

图 4　减速器设计问题

Fig. 4　Speed reducer design problem

表 6　减速器设计问题的求解结果

Table 6　Numerical results of speed reducer design problem

算法	最优值	平均值	最差值	标准差
ε CDOA	2 994.341 316	2 994.341 316	2 994.341 316	9.25×10^{-13}
ε DOA	2 994.449 254	3 000.635 591	3 010.655 669	5.26
PSO-DE	2 996.348 167	2 996.348 174	2 996.348 204	6.40×10^{-6}
DEDS	2 994.471 066	2 994.471 066	2 994.471 066	3.60×10^{-12}
DELC	2 994.471 066	2 994.471 066	2 994.471 066	1.90×10^{-12}
ABC	2 997.058 412	2 997.058 412	NA	0
UABC	2 994.471 066	2 994.471 072	NA	5.98×10^{-6}

注:NA 为缺失或无效的数据。

6　结论

(1)为了求解约束优化问题,提出了一种结合 ε 约束处理法和纵横交叉策略的野狗优化算法以解决工程设计问题。首先在野狗优化算法所有个体选择捕猎策略更新位置后引入纵横交叉策略,使所

得算法的全局搜索能力和跳出局部最优的能力得到提高;然后按照 ε 约束处理法,通过自适应的 ε 值将约束违约度 $\leqslant \varepsilon$ 的不可行解视为可行解,并用 ε 水平比较方法评判算法中野狗个体的优劣;最后根据约束违约度将种群分为两个子群,不同子群中的个体按不同的存活策略计算存活率,并更新两个子群中存活率较低的个体。在 19 个标准约束优化问题上进行数值实验并与相关算法的数值结果进行比较,结果表明算法 εCDOA 在求解约束优化问题时有令人满意的数值效果。将算法 εCDOA 应用到压力容器设计问题、焊接梁设计问题和减速器设计问题,均能得到更好的设计方案。

(2)今后可以将算法 εCDOA 应用在数学模型可表示为约束优化问题的实际问题中,如电力系统经济调度、多种工程设计问题等。

参 考 文 献

[1] Rao S S. Engineering optimization: theory and practice[M]. Hoboken: John Wiley & Sons, 2019.

[2] Martins J R R A, Ning A. Engineering design optimization[M]. Cambridge: Cambridge University Press, 2021.

[3] Chang K H. Design theory and methods using CAD/CAE: the computer aided engineering design series[M]. New York: Academic Press, 2014.

[4] 李智勇, 黄滔, 陈少淼, 等. 约束优化进化算法综述[J]. 软件学报, 2017, 28(6): 1529-1546.
Li Zhiyong, Huang Tao, Chen Shaomiao, et al. Overview of constrained optimization evolutionary algorithms[J]. Journal of Software, 2017, 28(6): 1529-1546.

[5] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

[6] Xue J, Shen B. A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.

[7] Gong W, Cai Z H, Liang D W. Adaptive ranking mutation operator based differential evolution for constrained optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(4): 716-727.

[8] Saha A, Datta R, Deb K. Hybrid gradient projection based genetic algorithms for constrained optimization[C]//The Proceedings of IEEE Congress on Evolutionary Computation. New York: IEEE, 2010: 2851-2858.

[9] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation, 2000, 4(3): 284-294.

[10] Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites[C]//The Proceedings of IEEE Congress on Evolutionary Computation. New York: IEEE, 2006: 372-378.

[11] Wang Y, Cai Z X, Guo G Q, et al. Multi-objective optimization and hybrid evolutionary algorithm to solve constrained optimization problems[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B(Cybernetics), 2007, 37(3): 560-575.

[12] Mallipeddi R, Suganthan P N. Ensemble of constraint handling techniques[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(4): 561-579.

[13] 方林, 蒋晓琳, 周庆丰, 等. 基于改进天鹰优化器的抱杆结构优化[J]. 科学技术与工程, 2023, 23(27): 11759-11767.
Fang Lin, Jiang Xiaolin, Zhou Qingfeng, et al. Structure optimization of holding pole based on improved aquila optimizer[J]. Science Technology and Engineering, 2023, 23(27): 11759-11767.

[14] 张美洲, 周敏, 黄静文, 等. 三空间分割遗传算法的环氧乙烷灭菌装载问题[J]. 科学技术与工程, 2022, 22(14): 5714-5722.
Zhang Meizhou, Zhou Min, Huang Jingwen, et al. Loading problem of ethylene oxide sterilization with three-space segmentation genetic algorithm[J]. Science Technology and Engineering, 2022, 22(14): 5714-5722.

[15] 王文川, 田维璨, 徐雷, 等. Mε-OIDE 求解约束优化问题算法及其在水库群防洪调度中的应用[J]. 水利学报, 2023, 54(2): 148-158.
Wang Wenchuan, Tian Weican, Xu Lei, et al. Mε-OIDE algorithm for solving constrained optimization problems and its application in flood control operation of reservoir group[J]. Journal of Hydraulic Engineering, 2023, 54(2): 148-158.

[16] Ji J Y, Tan Z, Zeng S, et al. An ε-constrained multiobjective differential evolution with adaptive gradient-based repair method for real-world constrained optimization problems[J]. Applied Soft Computing, 2024, 152. DOI: 10.1016/j.asoc.2023.111202.

[17] Deb K, Datta R. A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach[C]//The Proceedings of IEEE Congress on Evolutionary Computation. New York: IEEE, 2010: 1-8.

[18] Peraza-Vázquez H, Peña-Delgado A F, Echavarría-Castillo G, et al. A bio-inspired method for engineering design optimization inspired by dingoes hunting strategies[J]. Mathematical Problems in Engineering, 2021, 2021: 1-19.

[19] Meng A, Chen Y, Yin H, et al. Crisscross optimization algorithm and its application[J]. Knowledge-Based Systems, 2014, 67: 218-229.

[20] Liang J J, Runarsson T P, Mezura-Montes E, et al. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization[J]. Journal of Applied Mechanics, 2006, 41(8): 8-31.

[21] Lu H, Chen W. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems[J]. Journal of Global Optimization, 2008, 41: 427-445.

[22] Kohli M, Arora S. Chaotic grey wolf optimization algorithm for constrained optimization problems[J]. Journal of Computational Design and Engineering, 2018, 5(4): 458-472.

[23] de Castro Rodrigues M, Guimarães S, de Lima B S L P. E-BRM: a constraint handling technique to solve optimization problems with evolutionary algorithms[J]. Applied Soft Computing, 2018, 72: 14-29.

[24] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[25] Long W, Liang X, Cai S, et al. An improved artificial bee colony with modified augmented Lagrangian for constrained optimization[J]. Soft Computing, 2018, 22: 4789-4810.