http://bhxb.buaa.edu.cn jbuaa@buaa.edu.cn
DOI: 10.13700/j.bh.1001-5965.2023.0404

基于双重随机扰动的人工大猩猩部队优化算法及工程应用

杜晓昕[1,2,*]，郝田茹[1]，王波[1,2]，王振飞[1]，张剑飞[1,2]，金梅[1,2]

(1. 齐齐哈尔大学 计算机与控制工程学院，齐齐哈尔 161006；
2. 齐齐哈尔大学 黑龙江省大数据网络安全检测分析重点实验室，齐齐哈尔 161006)

摘 要：针对人工大猩猩部队优化算法(GTO)存在易陷入局部最优、收敛速度慢、寻优精度低等问题，提出了基于双重随机扰动策略的人工大猩猩部队优化算法(DGTO)。引入Halton序列初始化种群，增加种群的多样性；在算法寻优阶段使用多维随机数策略，并在探索阶段提出自适应位置搜索机制，提高算法的收敛速度；提出双重随机扰动策略，解决大猩猩的群居效应，增强算法跳出局部最优的能力；采用逐维更新策略更新个体位置，提升算法的收敛精度。通过10个基准测试函数寻优结果及Wilcoxon秩和检验对比可知，改进算法在寻优精度、收敛速度上有较大提升。同时，通过工程优化问题的实验对比分析，进一步验证了改进算法在处理现实工程问题上的优越性。

关 键 词：人工大猩猩部队优化算法；Halton序列；自适应位置搜索；双重随机扰动策略；逐维更新

中图分类号：TP301.6

文献标志码：A **文章编号**：1001-5965(2025)06-1882-15

群智能优化算法具有参数少、简单易实现、不需要梯度信息等优点，通过模拟自然界中动植物的求偶、觅食等行为，可以很好地在合理的时间且高度复杂的约束条件下找到这些问题的最优解。经典的群智能优化算法有粒子群(particle swarm optimization, PSO)算法[1]、人工蜂群(artificial bee colony, ABC)算法[2]、蚁群(ant colony optimization, ACO)算法[3]等。随着技术的不断发展和更新，群智能优化算法在旅行商[4]、支持向量机优化[5]、路线规划[6]、电力系统控制[7]等问题中有着卓越表现。

人工大猩猩部队优化算法(artificial gorilla troops optimizer, GTO)[8]是由Abdollahzadeh等于2021年提出的一种新型元启发式优化算法，通过对自然界中大猩猩个体觅食及争夺配偶行为的模拟而构造的一种有效的优化方案，虽然该算法具有原理简单易实现、所需调节参数少等优点，但依然存在易陷入局部最优、收敛速度慢等不足。自该算法提出以来，国内外不少学者针对其不足之处做出了改进。例如，Xiao等[9]提出了基于镜像对立和自适应-β爬山算法的大猩猩优化算法，采用凸透镜成像反向学习扩大搜索范围，避免陷入局部最优，同时引入自适应-β爬山算法与GTO结合以提高求解精度；Liang等[10]提出了一种基于反向学习和并行策略的GTO算法，通过反向学习拓展算法的探索空间，提高算法的全局搜索能力，并行策略将种群分为多个群体进行探索，增加了种群的多样性；Wu等[11]利用基于二次插值的天牛须搜索增强银背大猩猩位置的多样性，引入教与学优化算法的教师阶段，

收稿日期：2023-06-25；录用日期：2023-09-11；网络出版时间：2023-10-20 11:49
网络出版地址：link.cnki.net/urlid/11.2625.V.20231020.0903.001
基金项目：黑龙江省省属高等学校基本科研业务费自然科学类青年创新人才项目(145209206)
*通信作者. E-mail：xiaoxindu@qqhru.edu.cn

引用格式：杜晓昕，郝田茹，王波，等. 基于双重随机扰动的人工大猩猩部队优化算法及工程应用[J]. 北京航空航天大学学报，2025，51(6)：1882-1896. DU X X，HAO T R，WANG B，et al. Artificial gorilla troops optimizer based on double random disturbance and its application of engineering problem[J]. Journal of Beijing University of Aeronautics and Astronautics，2025，51(6)：1882-1896 (in Chinese).

以 50% 的概率更新跟随银背的行为, 并由准反射学习生成银背大猩猩的准反射位置; Alsolai 等[12] 提出增强型大猩猩, 利用 circle 混沌映射增加大猩猩的种群多样性, 并将其应用于无人机辅助智能车载网络中的分簇协议中; Mostafa 等[13] 引入精英反向学习增强种群的多样性, 利用柯西逆累积分布算子和切线飞行的融合提高种群的开发能力和收敛速度。尽管上述研究对 GTO 算法的寻优精度和速度有了一定的提升, 但该算法提出时间较短, 改进方法还不够完善, 仍然存在以下不足: ①迭代更新前的初始种群依赖初始条件较多, 健壮性不足; ②种群个体迭代更新策略较为单一, 且使用方式较为机械, 没有合理地平衡全局搜索与局部开发的关系; ③搜索过程中易掉入局部最优陷阱, 导致算法收敛精度不高; ④对于种群个体好坏的评价指标多样性差, 仅根据适应度函数的好坏便否决了种群个体的质量; ⑤种群个体位置更新方式不够细化。因此, 对于 GTO 算法的改进还需要更深入的研究。

本文提出基于双重随机扰动策略的人工大猩猩部队优化算法 (artificial gorilla troops optimizer based on double random disturbance strategy, DGTO) 。首先, 通过 Halton 序列生成均匀分布的初始种群, 增加初始种群的多样性; 其次, 在算法寻优阶段使用多维随机数策略, 并提出自适应位置搜索机制替代向已知位置搜索机制, 提高了算法的收敛速度; 然后, 提出一种双重随机扰动策略, 解决了大猩猩的群居效应, 避免算法过早收敛; 最后, 采用逐维更新策略更新个体位置, 提升了算法的收敛精度。通过对 10 个基准测试函数、Wilcoxon 秩和检验进行分析, 验证了本文算法的优越性, 并通过工程应用问题的实验结果分析, 验证了本文算法较其他算法更具有可行性。

1　人工大猩猩部队优化算法

GTO 算法是根据大猩猩的集体生活方式和社交方式提出的一种群智能优化算法。大猩猩生活在一个称为“部队”的群体中, 由成年雄性或银背大猩猩群体和几只成年雌性大猩猩及其后代组成, 银背大猩猩是群体的核心, 它做出所有决定, 调解战斗, 确定群体行动, 指导大猩猩寻找食物来源, 并为群体的安全福祉负责。在 GTO 算法中, 有 5 种不同的算子用于模拟大猩猩的群体行为, 主要分为 2 个阶段: 探索阶段和开发阶段。探索阶段采用 3 种不同的机制, 即迁移到未知位置、迁移到其他大猩猩位置及迁移到已知位置; 开发阶段采用跟随银背大猩猩和竞争成年雌性大猩猩 2 种社会行为。

大猩猩在探索阶段的位置更新方式如下:

$$G(t+1)=\begin{cases}(u_{\mathrm{b}}-l_{\mathrm{b}})r_1+l_{\mathrm{b}} & r<s\\ (r_2-C)X_{\mathrm{r}}(t)+LH & r\geqslant 0.5\\ X(t)-L[L(X(t)-G_{\mathrm{r}}(t))+r_3(X(t)-G_{\mathrm{r}}(t))] & r<0.5\end{cases}\tag{1}$$

式中: $G(t+1)$ 为下一次迭代中的大猩猩候选位置; $X(t)$ 为大猩猩的当前位置; r_1、r_2、r_3 和 r 表示取值为 (0,1) 的随机值; s 为优化操作前给定值的参数, 一般取 0.03, 其决定了大猩猩迁移到未知位置的概率; u_{b} 和 l_{b} 分别为变量的上限和下限; X_{r} 和 G_{r} 分别为从当前位置和候选位置中随机选择的大猩猩个体位置; C、L、H 分别表示为

$$C=(\cos(2r_4)+1)\left(1-\frac{I_t}{I_{t,\max}}\right)\tag{2}$$

$$L=Cl\tag{3}$$

$$H=ZX(t)\tag{4}$$

其中: I_t 为当前迭代值; $I_{t,\max}$ 为执行优化操作的迭代总值; r_4 表示取值为 (0,1) 的随机值; l 表示取值为 (−1,1) 的随机值; Z 表示问题维度中取值为 $[-C,C]$ 的随机值。

大猩猩在开发阶段选择跟随银背大猩猩机制, 其行为表示为

$$G(t+1)=LM(X(t)-X_{\mathrm{silverback}})+X(t)\tag{5}$$

$$M=\left(\left|\frac{1}{N}\sum_{i=1}^{N}G_i(t)\right|^{g}\right)^{\frac{1}{g}}\tag{6}$$

$$g=2^L\tag{7}$$

式中: $X_{\mathrm{silverback}}$ 为银背大猩猩 (最佳解) 的位置; $G_i(t)$ 为第 i 个候选大猩猩在第 t 次迭代时的位置; N 为大猩猩的总数。

大猩猩在开发阶段选择竞争成年雌性大猩猩机制, 其行为表示为

$$G(t+1)=X_{\mathrm{silverback}}-(X_{\mathrm{silverback}}Q-X(t)Q)A\tag{8}$$

$$Q=2r_5-1\tag{9}$$

$$A=\beta E\tag{10}$$

$$E=\begin{cases}N_1 & r\geqslant 0.5\\ N_2 & r<0.5\end{cases}\tag{11}$$

式中: Q 用于模拟冲击力; A 为冲突中暴力程度的系数; r_5 表示取值为 (0,1) 的随机值; β 为优化操作前给定的参数值; E 用于模拟暴力对解的维度的影响, 如果 $r\geqslant 0.5$, E 等于正态分布和问题维度中的随

机值 N_1，反之，E 等于正态分布中的随机值 N_2。

2 改进的人工大猩猩部队优化算法

在标准 GTO 算法中，种群初始化采用随机分布的方式，这种分布方式使得种群分布极不均匀，进而导致算法收敛速度缓慢；参数 C 随机生成时未考虑问题维度，使得到的种群多样性不高，同时，探索阶段的向已知位置迁移机制也未能很好地驱使种群迁移到更好的位置；种群跟随银背大猩猩和普通成年大猩猩进行位置更新，当种群个体陷入局部极值时，算法会出现停滞搜索现象；在更新个体位置时，算法中的种群个体在进行位置更新时，并未考虑到每一维度的位置信息，而是选择整体更新的方式，导致算法收敛精度变低。

本文针对标准 GTO 算法的不足之处，提出相应的策略对其进行改善。

2.1 Halton 序列初始化种群

种群初始化是群智能优化算法不可或缺的阶段，传统的种群初始化方法依赖伪随机数生成，该方式实现简单且随机性强，但种群个体不一定均匀地分布在搜索空间中[14]，从而使种群个体错过相当大的一部分种群空间。基于混沌技术生成初始化种群的方式也被广泛应用，已有研究表明，利用混沌序列进行种群初始化能取得比伪随机数更好的效果，但该方式整体稳定而局部不稳定，且对初始条件及其参数设置非常敏感。此外，多步骤技术初始化种群也较为流行，通常是在后续步骤中细化先前生成的种群，其中，最流行的是基于反向学习的技术。首先，产生原始群体的点；然后，使用反向学习技术对生成点中不符合条件的点进行反向学习，这种方式虽然可以取得良好的效果，但会受到第一步生成的种群点质量的影响，且需要消耗一部分计算预算来评估适应度函数。基于此，本文引入较为先进的准随机序列方式，即 Halton 序列来初始化种群。Halton 序列是一种低差异序列，以质数为基数的确定性方法构造，使得种群更加均匀地分布在整个解空间，增加了种群的多样性，从而加快了算法的收敛速度，并提高了算法的收敛精度。

假设搜索空间为二维，Halton 序列的实现过程为：选取 2 个质数作为基数，分别对应 2 个维度，每个维度根据基数分别对 (0,1) 不断地进行切分、迂回取值，从而形成不重复且均匀的点，其切分过程的数学模型为

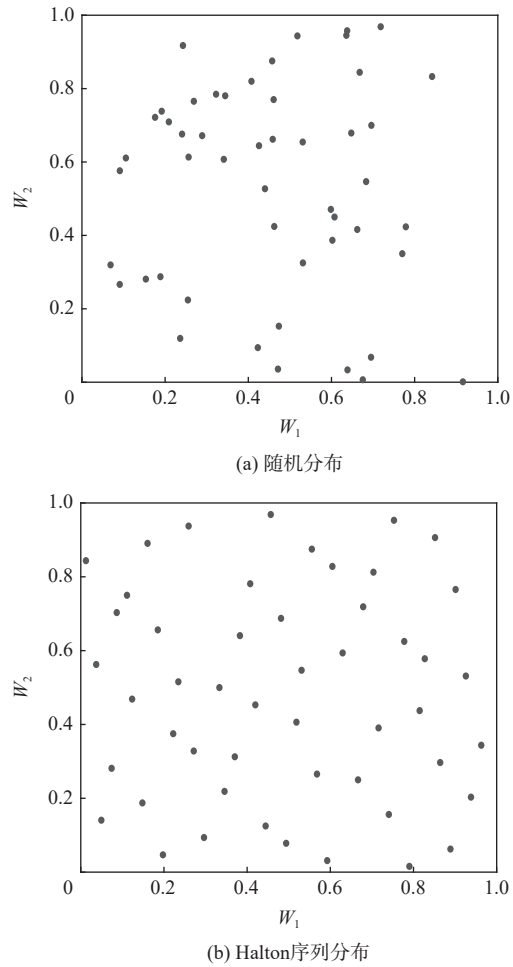
$$n=\sum_{i=0}^{m_0}a_i p_1^i=a_0+a_1p_1^1+\cdots+a_{m_0}p_1^{m_0} \tag{12}$$

$$\phi_{p_1}(n)=a_0p_1^{-1}+a_1p_1^{-2}+\cdots+a_{m_0}p_1^{-m_0-1} \tag{13}$$

$$H(n)=\{\phi_{p_{1,1}}(n),\phi_{p_{1,2}}(n)\} \tag{14}$$

式中：n 为 Halton 序列的序数；p_1 为 Halton 序列的基数，取值为大于或等于 2 的质数；m_0 为非负整数，用于确定求和的项数上限；$a_i\in\{0,1,2,\cdots,p_1-1\}$为常变量；$\phi_{p_1}(n)$为定义的序列函数；$H(n)$ 为二维均匀 Halton 序列。

在二维空间下，假设种群规模为 50，个体分布范围的上下界分别为 1 和 0，随机分布和 Halton 序列分布初始化种群空间个体分布对比如图 1 所示，其中，Halton 序列的基数分别为 2 和 3，W_1 和 W_2 为维度。由图 1 对比可知，通过 Halton 序列初始化种群空间的个体分布虽然降低了随机性，但分布更加均匀，增加了算法的多样性。



(a) 随机分布

(b) Halton序列分布

图 1 随机分布与 Halton 序列分布初始化种群空间

Fig. 1 Random distribution and Halton distribution initialize population space

2.2 自适应位置搜索机制

在标准 GTO 算法中，大猩猩探索阶段寻优时使用了 3 种不同的机制，即迁移到未知位置、迁移到其他大猩猩位置和向已知位置迁移。然而，原始的向已知位置迁移机制过多考虑大猩猩个体自身搜索的经验，没有自适应地平衡好大猩猩个体勘探

与开发之间的关系。针对群智能优化算法中普遍存在的缺乏自适应调整能力的问题, 出现了多种基于欧氏距离、适应度函数、迭代次数的自适应步长方法。文献 [15] 使用基于欧氏距离的自适应步长控制麻雀加入者和发现者各维度之间的距离, 当加入者和发现者最优位置较远时, 步长较大, 提高算法收敛速度, 反之, 减小步长, 提高算法的局部搜索能力; 文献 [16] 将果蝇种群每次寻优得到的适应度函数最好的全局最优解融入到寻优迭代的过程中, 避免果蝇个体盲目搜索; 文献 [17] 使用基于迭代次数的自适应步长方法优化步长因子, 使其随迭代次数增加不断减小。为使 GTO 算法能够自适应平衡全局搜索与局部开发之间的关系, 本文通过分别调整个体自身位置对迭代影响的自适应因子及其他大猩猩个体对本个体搜索影响的自适应因子, 提出一种自适应位置搜索机制替代该搜索机制, 具体表达式为

$$G(t+1)=TX(t)-r_6L(X(t)-G_{\mathrm{r}}(t)) \tag{15}$$

$$T=\left(\frac{1}{1+\mathrm{e}^{K}}\right)^{4} \tag{16}$$

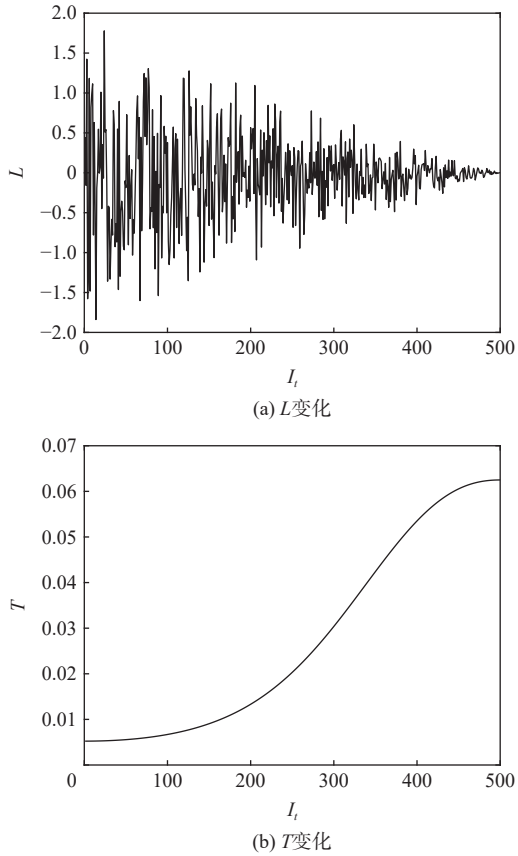
式中: L 的取值受 C 影响, 为使其考虑问题维度, 对 C 采用了多维随机数策略; r_6 表示取值为 (0,1) 的随机值; $K=\left(1-\dfrac{I_t}{I_{t,\max}}\right)^{2I_t/I_{t,\max}}$。

L 和 T 随迭代次数增加的变化曲线如图 2 所示。可知, 大猩猩个体自身位置的权重 T 取值从 0.005 不断上升到 0.063, 而为大猩猩个体受其他已知位置的影响部分 $X(t)-G_{\mathrm{r}}(t)$ 赋予权重 L, 其取值从 [−2,2] 不断向 0 靠近。在前 200 次迭代中, T 增长平缓, L 取值较大, 个体更注重其他位置的探索; 在 200～400 次迭代中, T 增长迅速, L 持续减小, 该阶段个体逐渐由探索其他位置转为开发自身邻域位置; 在最后 100 次迭代中, T 趋于平稳, 而 L 也逐渐稳定到 0 附近, 此时个体更注重自身邻域的搜索。

相比原始迁移到已知位置机制, 本文提出的自适应位置搜索机制更符合个体前期精于探索、后期精于开发的搜索模式, 进一步提高了算法的收敛速度。

2.3　双重随机扰动策略

大猩猩在社会生活中倾向于集体生活, 它们作为一个群体寻找食物, 并生活在银背大猩猩的领导下, 听从银背大猩猩的指挥决策。虽然大猩猩的集体生活方式通过个体之间的信息交流使得优质食物源更容易被个体搜寻到, 但也会因此产生潜在的群居效应, 即种群在找到自身认为是最优食物源后把它作为全局最好的食物源, 从而停止对更好食物源的探索。为减小群居效应对种群寻优产生的影响, 本文受元学习[18] 策略的启发, 通过对种群个体适应度值及适应度变化率的评估, 提出一种双重随机扰动策略帮助种群在完成位置迭代更新后跳出局部最优值。元学习策略即“学会学习的学习”, 即使使用少量样本数据, 也能快速学会新的概念或技能。元学习的样本是基于任务的, 其是从任务中学习特征表示, 从而实现在新任务上的泛化。然而, 各学习方法并不是单独存在的, 与其他学习方法存在潜在的联系。表 1 展示了元学习与其他学习方法之间的潜在关联。在双重随机扰动策略中, 寻找全局最优解过程中跳出局部最优解是种群个体的任务, 个体适应度值是否比适应度平均值低,



(a) L变化

(b) T变化

图 2　L 和 T 随迭代次数增加的变化曲线

Fig. 2　L and T change curves with increase of iterations

表 1　元学习与其他学习方法的潜在关联

Table 1　Potential correlation between meta-learning and other learning methods

学习方法	与元学习关联
对比学习	对比学习的核心是将正样本和负样本在特征空间进行对比, 使得模型能够学习到样本的重要内在特征。而元学习在学习特征表示时, 有时也会通过正样本和负样本比对的方式实现
迁移学习	迁移学习的核心是找到已有知识和新知识之间的相似性, 通过这种相似性的迁移达到迁移学习的目的。元学习基于任务展开学习, 然而生物体及其一生, 学习的永远不止一个任务, 因而迁移学习可以将元学习在某一任务的学习方式迁移到相似任务中

以及个体适应度变化率是否比优质个体适应度变化率大，即为种群个体的特征表示，个体根据自身特征选择不同的对策，双重随机扰动得以实现。由表1可知，双重随机扰动策略在元学习过程中，个体适应度值与适应度平均值的比较及个体适应度变化率与优质个体适应度变化率的比较结合了对比学习的核心思想，同时也提升了评判个体质量的多样性；而个体在双重随机扰动过程中学习到的如何跳出局部最优值的能力，也可以通过迁移学习迁移到更多相似的问题中。因此，该策略有良好的泛化性。双重随机扰动策略如图3所示，详细步骤如下：

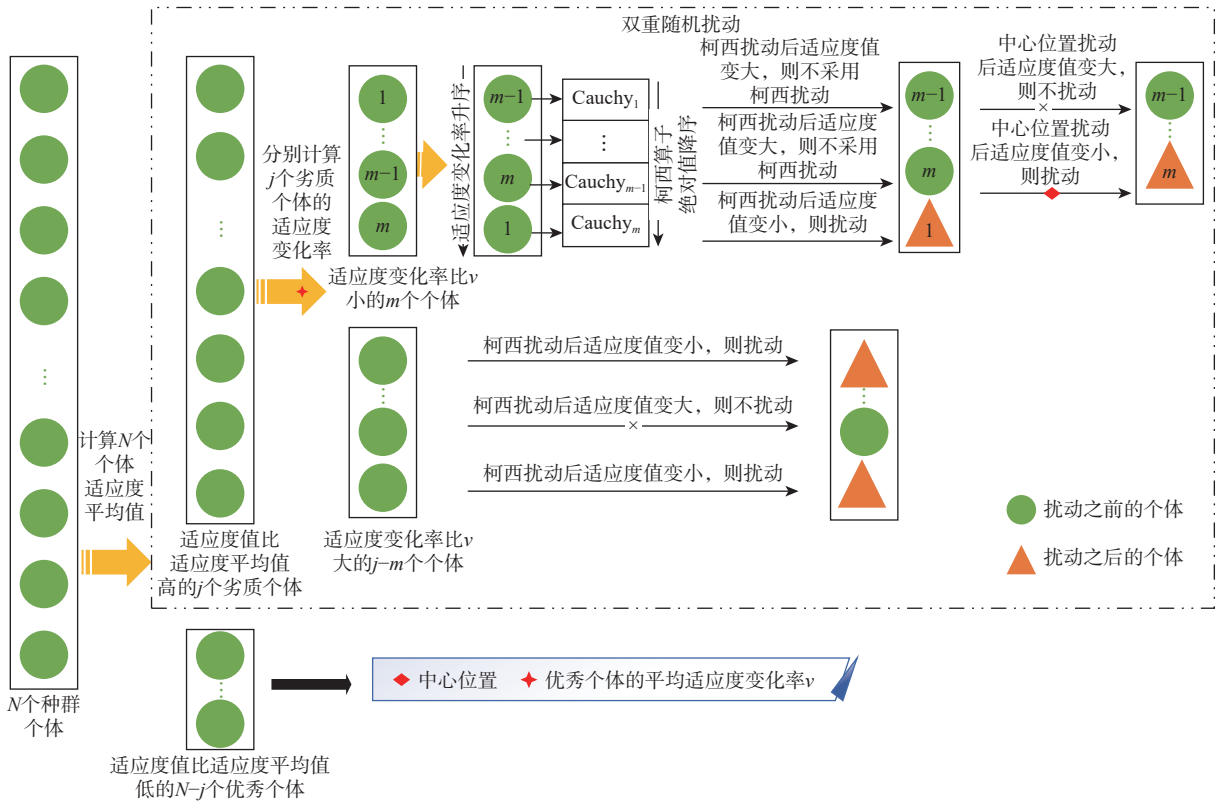


图3 双重随机扰动策略示意图

Fig. 3 Schematic diagram of double random disturbance strategy

步骤1 根据式(17)计算每次迭代后 N 个大猩猩个体的适应度平均值：

$$f_{\mathrm{v}}=\frac{\sum_{i=1}^{N} f_i}{N} \tag{17}$$

式中：f_i 为个体的适应度值。

步骤2 根据适应度平均值将种群个体分为 j 个比适应度平均值高的劣质个体和 $N-j$ 个比适应度平均值低的优质个体。

步骤3 根据式(18)分别计算 j 个劣质个体和 $N-j$ 个优质个体的适应度变化率 v_1，并求取优质个体的平均适应度变化率 v。

$$v_1=\frac{\Delta f}{\Delta I_t} \tag{18}$$

式中：Δf 为适应度的变化大小；ΔI_t 为迭代次数的变化大小。

步骤4 双重随机扰动策略。

1）第一重是柯西扰动[19]。一维标准柯西分布的概率密度函数表达式为

$$f(x)=\frac{1}{\pi}\left(\frac{1}{x^2+1}\right) \qquad -\infty<x<\infty \tag{19}$$

一维标准柯西分布与一维标准高斯分布概率密度曲线如图4所示。可知，柯西分布较高斯分布

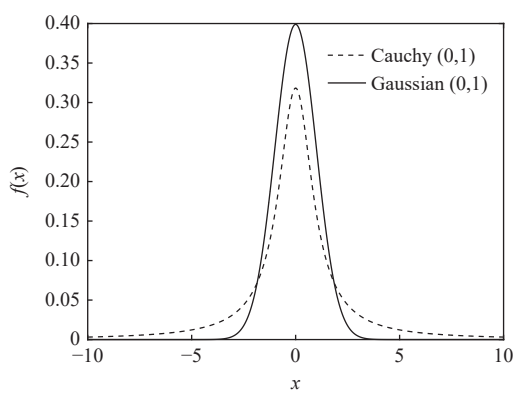


图4 一维标准柯西分布与高斯分布概率密度曲线

Fig. 4 Probability density curves of one-dimensional standard Cauchy distribution and Gaussian distribution

在原点处的峰值更小, 使得个体变异后使用较少的时间搜索相邻区域, 同时, 柯西分布相较高斯分布具有长尾特性, 使其有一定概率取到距离原点较远的随机数, 增加了全局搜索的可能性。这意味着经过柯西变异后的个体具有跳出局部最优值的能力, 其扰动方式如下:

$$G = X - K \cdot \text{Cauchy}(0,1) \tag{20}$$

式中: K 用来决定柯西算子扰动的作用大小; Cauchy(0,1) 为标准柯西分布。柯西分布随机变量生成函数为 $\eta = \tan((\xi - 0.5)\pi)$, ξ 为 [0,1] 上服从均匀分布的随机变量。

2) 第二重是中心位置扰动策略。先计算 N–j 个优质个体的中心位置, 再使用该位置对劣质个体进行扰动。中心位置 O 的计算如图 5 所示。图中: W 为个体位置的维度值。中心位置代表优质个体的平均水平, 该扰动策略促使陷入局部最优值的个体能从当前位置向优质个体的中心位置移动, 扰动方式如下:

$$G = X - r\,(X - O) \tag{21}$$

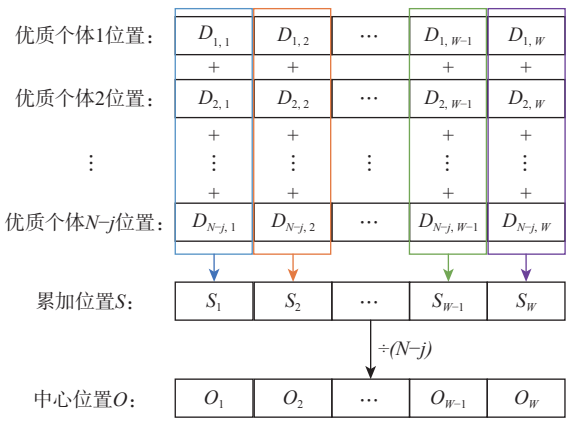


图 5　中心位置计算示意图

Fig. 5　Schematic diagram of central location calculation

步骤 5　对 j 个劣质个体进行分类。根据步骤 3 计算的适应度变化率将 j 个劣质个体分为 2 类: ①适应度变化率比优质个体平均适应度变化率低的 m 个个体; ②适应度变化率比优质个体平均适应度变化率高的 j–m 个个体。

步骤 6　对分类后的 j 个劣质个体进行双重随机扰动。第 1 类个体代表适应度达不到平均水平, 寻优速度也比优质个体平均寻优速度慢。对这类劣质个体进行第一重柯西扰动, 将第 1 类 j 个劣质个体按照适应度变化率进行排序, 适应度变化率小的个体较其他个体有更大的几率陷入局部最优值, 因此, 更需要大的步长对其进行扰动。同时, 生成 j 个柯西算子, 对其按照绝对值大小进行排序, 柯西算子绝对值越大, 代表扰动的步长越大。将适应度变化率较小的个体与绝对值较大的柯西算子(即大步长)一一对应, 利用式 (20) 对其进行扰动, 若扰动后适应度值变小, 则保留扰动后的位置, 否则, 根据式 (21) 启用第二重中心位置扰动。同样, 若扰动后适应度值变小, 则保留扰动后的位置, 否则, 不扰动。第 2 类个体代表适应度不及平均水平, 但寻优速度比优质个体的平均寻优速度快, 因此, 仅对其采用第一重柯西扰动, 若扰动后适应度值变小, 则保留扰动后的位置, 否则, 不扰动。

在个体完成位置更新后, 对个体进行双重随机扰动, 帮助个体减小陷入潜在局部最优的风险, 促使个体更好地向全局最优解的位置探索。

2.4　逐维更新

标准 GTO 算法在更新个体位置比较适应度时, 将种群个体的不同维度视为一个整体进行评估[20], 但某一维度上较好的结果可能会被忽略, 从而降低了算法的收敛速度和求解精度。因此, 本文引入一种逐维更新的策略, 用于更新大猩猩的位置, 该策略使得每一维的信息都得到关注。在大猩猩位置更新过程中, 每更新一个维度上的值, 就与其他维度的值组成一个完整的解, 再评价这个新的解, 直到每一维的值都被组合比较。

标准 GTO 算法位置更新方式与逐维更新位置方式如图 6 和图 7 所示。图中: D_{Xi}、D_{Gi} 分别为大猩猩在当前位置 X、候选位置 G 第 i 维度的位置信息, $f(X)$、$f(G)$ 表示适应度值。

通过引入逐维更新策略, 个体每一维的位置都得到关注, 促使每一维好的位置信息对寻优发挥一定的帮助, 也有效防止了每一维不好的位置信息对寻优结果产生坏的影响, 从而提高算法的寻优精度。

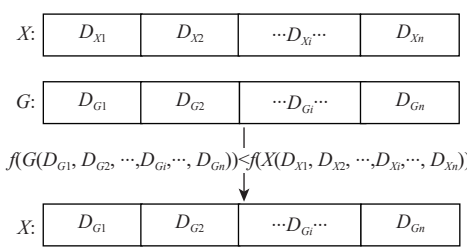


图 6　整体更新位置方式

Fig. 6　Entire update position way

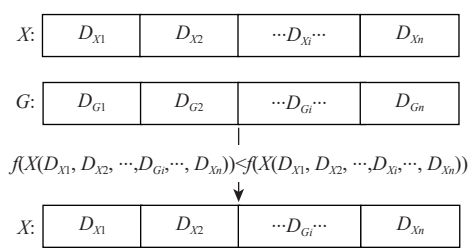


图 7　逐维更新位置方式

Fig. 7　Dimension-by-dimension update position way

2.5 算法实现流程

本文提出的DGTO算法步骤如下，其流程如图8所示。

步骤1 初始化相关参数：种群规模N、维度W、最大迭代次数$I_{t,\max}$、种群搜索范围u_b和l_b、探索阶段切换概率s、开发阶段切换概率w、参数β。

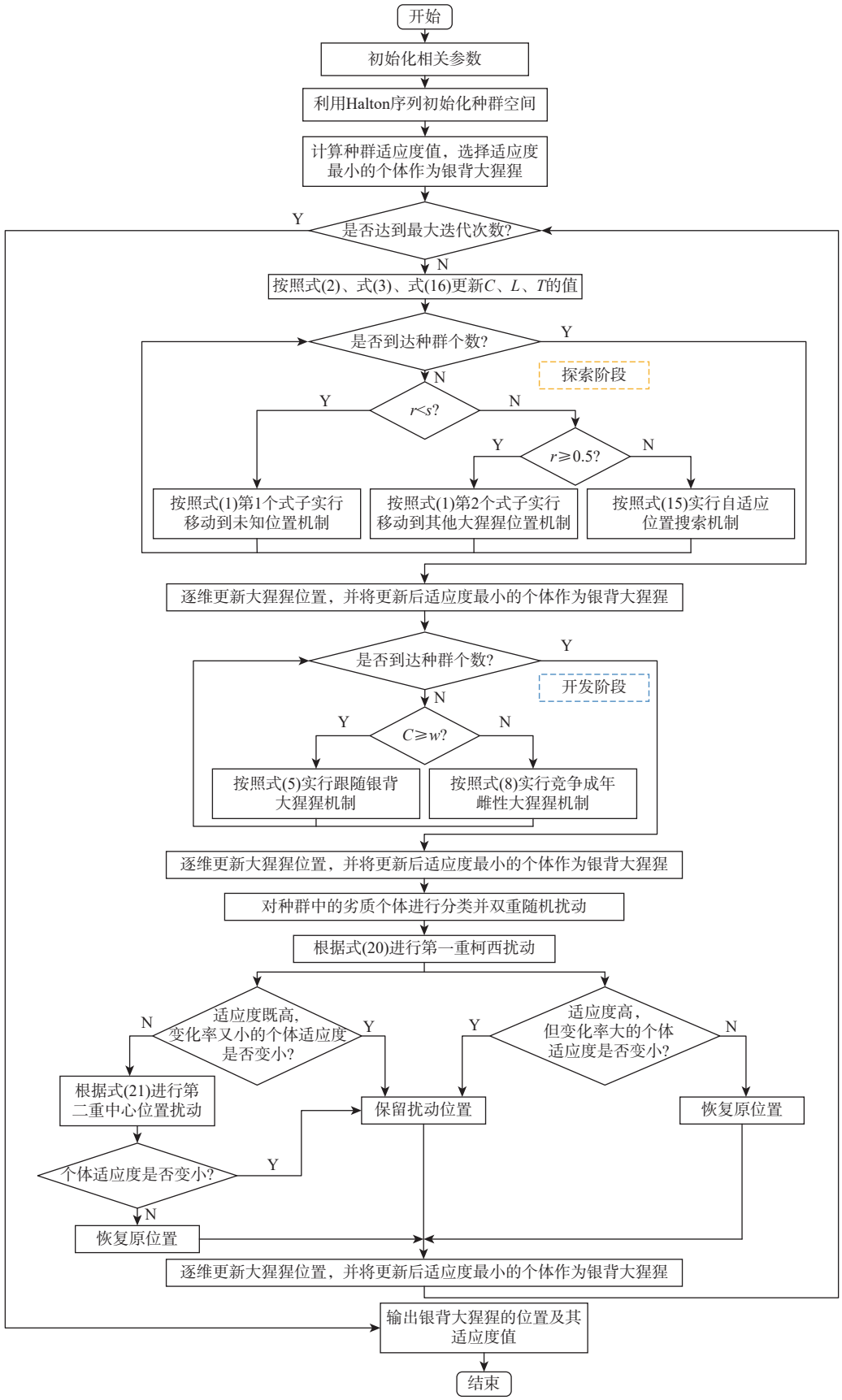


图8 本文算法流程

Fig. 8 Flowchart of the proposed algorithm

步骤 2　利用 Halton 序列初始化种群 X。

步骤 3　计算每个大猩猩个体 X 的适应度值, 并选择适应度最小的个体作为银背大猩猩 $X_{\text{silverback}}$。

步骤 4　按照式 (2)、式 (3)、式 (16) 计算参数 C、L、T 的值。

步骤 5　按照式 (1) 第 1 个式子、第 2 个式子及式 (15)、探索阶段切换概率 s 更新大猩猩候选位置 G, 并对更新后的候选位置 G 进行边界检查, 根据逐维更新策略更新大猩猩的位置 X, 选择适应度最小的大猩猩个体作为银背大猩猩 $X_{\text{silverback}}$。

步骤 6　按照式 (5)、式 (8) 及开发阶段切换概率 w 更新大猩猩候选位置 G, 并对更新后的位置 G 进行边界检查, 根据逐维更新策略更新大猩猩的位置 X, 选择适应度更小的大猩猩个体作为银背大猩猩 $X_{\text{silverback}}$。

步骤 7　按照双重随机扰动策略及式 (20)、式 (21) 对有可能陷入局部最优的个体进行位置扰动, 再根据逐维更新策略更新大猩猩的位置 X, 选择适应度更小的大猩猩个体作为银背大猩猩 $X_{\text{silverback}}$。

步骤 8　判断是否满足迭代终止条件, 满足, 则输出银背大猩猩位置 $X_{\text{silverback}}$ 及其适应度值, 否则, 进入步骤 4 继续执行。

3　实验仿真与结果分析

3.1　参数设置说明

基于 AMD Ryzen 75800H with Radeon Graphics CPU、3.20 GHz 主频、16 GB 内存及 Windows 11 (64 位) 的操作系统对 DGTO 算法进行仿真实验。编程软件为 MATLAB R2019a。选取标准 GTO 算法、灰狼优化算法 (grey wolf optimizer, GWO)[21]、差分进化 (differential evolution, DE) 算法[22]、鲸鱼优化算法 (whale optimization algorithm, WOA)[23] 与 DGTO 算法作对比, 基本参数统一设置为: 种群规模为 40, 最大迭代次数为 500, 维度为 10。各算法内部参数如表 2 所示。表中: F 为缩放因子; R_C 为交叉概率; b 为螺旋形状控制参数。

表 2　参数设置

Table 2　Parameters setting

算法	主要参数
GTO	s=0.03,w=0.8,β=3
GWO	
DE	F=0.5,R_C=0.3
WOA	b=1
DGTO	s=0.03,w=0.8,β=3

3.2　测试函数

为测试 DGTO 算法的寻优性能, 从基准测试函数中选取 10 个进行寻优测试, 其中, f_1～f_6 为单峰测试函数, f_7、f_8 为多峰测试函数, f_9、f_{10} 为固定维多峰测试函数。基准测试函数相关信息如表 3 所示。

表 3　基准测试函数

Table 3　Benchmark test functions

编号	函数名	定义域	维度	最优值
f_1	Sphere	[−100,100]	10/30	0
f_2	Schwefel' problem 2.22	[−10,10]	10/30	0
f_3	Schwefel' problem 1.2	[−100,100]	10/30	0
f_4	Schwefel' problem 2.21	[−100,100]	10/30	0
f_5	Generalized Rosenbrock's function	[−30,30]	10/30	0
f_6	Step function	[−100,100]	10/30	0
f_7	Generalized Schwefel's problem 2.26	[−500,500]	10/30	−418.98×维度
f_8	Generalized penalized function 2	[−50,50]	10/30	0
f_9	Kowalik's function	[−5,5]	4	0.000 3
f_{10}	Hatman's function 2	[0,1]	6	−3.32

3.3　DGTO 算法与其他改进 GTO 算法的性能对比及收敛性分析

为验证 DGTO 算法改进的有效性, 将其与标准 GTO 算法及基于镜像对立和自适应-β爬山算法改进的 GTO 算法 (improved gorilla troops optimizer based on lens opposition-based learning and adaptive β-hill climbing, IGTO)、用于全局优化问题的改进 GTO 算法 (modified gorilla troops optimizer for global optimization problem, MGTO) 在表 3 所示的基准测试函数上进行测试对比, IGTO 和 MGTO 的数据分别来自文献 [9] 和文献 [11]。为保证实验的可比性, 实验统一设置为: 种群规模为 30, 维度为 30, 最大迭代次数为 500。各算法分别运行 30 次取平均值和标准差, 对比结果如表 4 所示, 加粗字体表示

表4 DGTO算法与其他改进GTO算法寻优结果对比

Table 4 Comparison of optimization results of DGTO and other improved GTO algorithms

参数	算法	平均值	标准差
f_1	GTO	0	0
	IGTO	0	0
	MGTO	0	0
	DGTO	**0**	**0**
f_2	GTO	6.67×10^{-195}	0
	IGTO	0	0
	MGTO	0	0
	DGTO	**0**	**0**
f_3	GTO	0	0
	IGTO	0	0
	MGTO	0	0
	DGTO	**0**	**0**
f_4	GTO	1.01×10^{-192}	0
	IGTO	0	0
	MGTO	0	0
	DGTO	**0**	**0**
f_5	GTO	3.17×10^{0}	8.22×10^{0}
	IGTO	7.46×10^{-5}	8.57×10^{-5}
	MGTO	2.54×10^{-5}	3.94×10^{-5}
	DGTO	$\mathbf{7.42\times10^{-6}}$	$\mathbf{3.39\times10^{-5}}$
f_6	GTO	2.56×10^{-7}	4.19×10^{-7}
	IGTO	1.01×10^{-7}	1.15×10^{-7}
	MGTO	3.37×10^{-14}	3.76×10^{-14}
	DGTO	$\mathbf{2.03\times10^{-32}}$	$\mathbf{1.66\times10^{-32}}$
f_7	GTO	-1.26×10^{4}	3.05×10^{-5}
	IGTO	-1.26×10^{4}	3.02×10^{-5}
	MGTO	-1.26×10^{4}	1.03×10^{-6}
	DGTO	$\mathbf{-1.26\times10^{4}}$	$\mathbf{4.64\times10^{-12}}$
f_8	GTO	3.30×10^{-3}	5.12×10^{-3}
	IGTO	1.14×10^{-7}	3.34×10^{-7}
	MGTO	1.00×10^{-7}	1.43×10^{-7}
	DGTO	$\mathbf{7.24\times10^{-32}}$	$\mathbf{2.30\times10^{-31}}$
f_9	GTO	4.17×10^{-4}	3.01×10^{-4}
	IGTO	3.07×10^{-4}	4.06×10^{-19}
	MGTO	3.07×10^{-4}	3.29×10^{-19}
	DGTO	$\mathbf{3.07\times10^{-4}}$	$\mathbf{1.74\times10^{-19}}$
f_{10}	GTO	-3.29×10^{0}	5.39×10^{-2}
	IGTO	-3.31×10^{0}	3.63×10^{-2}
	MGTO	-3.30×10^{0}	4.51×10^{-2}
	DGTO	$\mathbf{-3.32\times10^{0}}$	**0**

在某一函数下各算法的最优结果。图9为各算法的部分收敛曲线。

由表4和图9可知，在10个基准测试函数中，DGTO算法较其他算法具有显著优势，其平均值、标准差大多具有量级优势。对于函数f_1～f_4，MGTO、IGTO算法可以找到理论最优值，稳定性良好，而DGTO算法通过不同的方式改进也达到了相同的效果，说明了DGTO算法的有效性，为算法改进提供了新的思路；对于函数f_5和f_{10}，DGTO算法的寻优结果最好，也最为稳定，f_5的收敛速度与其余算法相比较为明显，f_{10}甚至达到了标准差为0的稳定性；对于函数f_6，GTO、IGTO算法的收敛精度基本相同，可推断其陷入局部最优值而无法自行跳出，而MGTO算法收敛精度略高，但DGTO算法在双重随机扰动策略的帮助下，具备了更强的跳出局部最优值的能力，达到了更高的收敛精度及更强的稳定性，从收敛曲线也可看出，在收敛次数约为50次时，DGTO算法已经达到了其余算法的精度值，在其余算法逐渐收敛后，DGTO算法依旧探索更好的结果；对于函数f_8，IGTO、MGTO算法虽比GTO算法平均值更低，但DGTO算法还是达到了约10^{-32}的精度和标准差，差距较为明显；对于函数f_7，各算法虽然都可寻到最优值附近，但DGTO算法的性能更为稳定；对于函数f_9，改进后的GTO算法比标准GTO算法更靠近理论最优值，而DGTO算法的收敛曲线下降速率更大，说明其找到理论最优值所需的迭代次数更少，且更稳定。综上所述，DGTO算法的优势较为明显。

3.4 DGTO算法与其他算法的性能对比

为验证DGTO算法对于基准测试函数的寻优性能，选取标准GTO算法、GWO算法、DE算法、WOA算法进行寻优对比。为确保实验结果具有可比性，各算法实验统一设置为：种群规模为40，维度为10，最大迭代次数为500，各算法相关参数由表2给出，利用表3的10个基准测试函数对各算法进行寻优性能对比，各算法分别运行50次取最优值、最差值、平均值和标准差，对比结果如表5所示，加粗字体表示在某一函数下各算法的最优结果。

由表5可知，在求解函数f_1～f_{10}时，DGTO算法寻优结果及寻优稳定性均优于其他算法。在多个函数中，DGTO算法的最差值都优于其余算法的最优值。对于单峰函数f_1～f_4，DGTO算法每次都可以直接收敛到理论最优值0；对于单峰函数f_5，DGTO算法较其他算法更接近理论最优值；对于单峰函数f_6，DGTO算法最优可以收敛到理论最优值0；对于多峰函数f_7和固定维多峰函数f_9、f_{10}，DGTO算法更接近理论最优值−4 189.8、0.000 3和−3.32；对于多峰函数f_8，DGTO算法可以收敛到理论最优值附近。综上所述，DGTO算法较其他算法在对基准测试函数进行求解时具有明显优势。

3.5 DGTO算法与其他算法的收敛性对比

为验证DGTO算法的收敛性，图10给出了DGTO算法、标准GTO算法、GWO算法、DE算

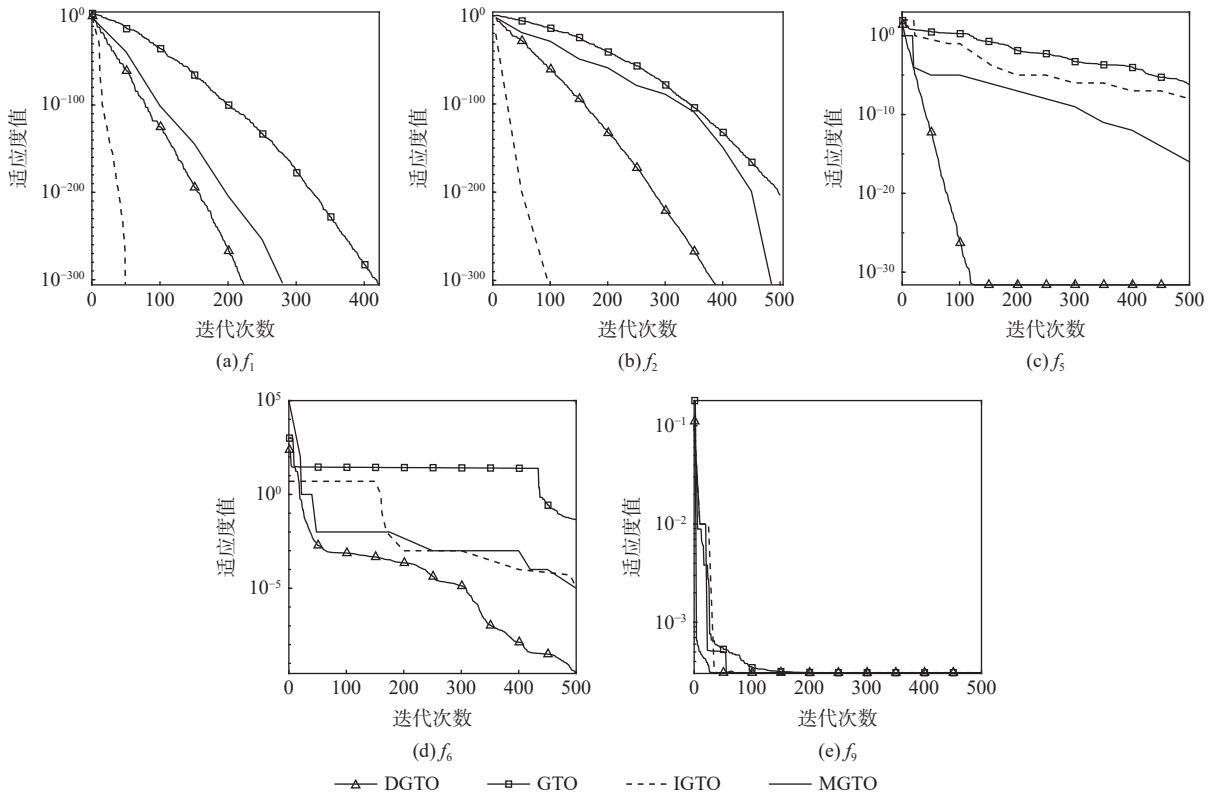


图 9　DGTO 算法与其他改进 GTO 算法的部分收敛曲线

Fig. 9　Partial convergence plots of DGTO and other improved GTO algorithms

法、WOA 算法在 10 个基本测试函数上的收敛曲线。各算法实验仍统一设置为: 种群规模为 40, 维度为 10, 最大迭代次数为 500。

由图 10 可知, 对于单峰函数 f_1～f_6 和多峰函数 f_7～f_{10}, 横向观察收敛曲线, DGTO 算法收敛曲线在整个寻优过程中均位于其余 4 个算法左方, 说明在初始迭代时, Halton 序列增加了种群的多样性, 随后自适应位置搜索机制合理地平衡了开发和探索之间的关系, 使得 DGTO 算法较其余 4 个算法有更好的收敛速度; 纵向观察收敛曲线, DGTO 收敛曲线在整个寻优过程中均位于其余 4 个算法下方, 说明双重随机扰动策略帮助算法跳出局部最优值, 最终逐维更新每一维度的信息进行比较更新, 使得 DGTO 算法较其他 4 个算法有更好的寻优精度。对于单峰函数 f_1、f_3, DGTO 算法分别仅需 208 次、252 次迭代便寻到理论最优值, 约为 GTO 算法寻优速度的 2 倍, 而 GWO、DE、WOA 算法经过 500 次迭代仍处于较低的寻优精度。对于函数 f_2、f_4, f_6, DGTO 算法分别在 380 次、398 次、102 次迭代找到了理论最优值, 而其余 4 个算法仍与理论最优值差距较大。对于单峰函数 f_5, DGTO 算法时而平缓时而下降, 说明算法在陷入局部最优时, 双重随机扰动策略可以帮助其跳出局部最优, 继而向全局最优解靠近。综上所述, DGTO 算法在对基准测试函数寻优时具有更快的收敛速度和更高的收敛精度, 说明了算法的有效性。

3.6　DGTO 算法与其他算法的平均绝对值误差比较

为评估 DGTO 算法与其他算法在求解 10 个基准测试函数时的寻优结果与理论最优值的差异情况, 使用平均绝对值误差[24] 对其进行排序比较。平均绝对值误差可以避免误差相互抵消的问题, 因而可以准确反映实际预测误差的大小。各算法实验统一设置为: 种群规模为 40, 维度为 10, 最大迭代次数为 500, 独立运行 50 次。平均绝对值误差的数学模型如式 (22) 所示, 各算法对 10 个基准函数的平均绝对值误差排序结果如表 6 所示。

$$Y=\frac{1}{n_1}\sum_{i=1}^{n_1}|\mu_i-\sigma_i| \tag{22}$$

式中: n_1 为选用基准函数的个数; μ_i 为算法运行 50 次寻得最优值的平均值; σ_i 为每个基准测试的理论最优值。

由表 6 可知, DGTO 算法的平均绝对值误差最小, 说明其寻优结果更接近函数的理论最优值, 收敛精度较其他算法更高, 进一步证明了算法的有效性, 较其他算法更有优势。

3.7　Wilcoxon 秩和检验

为进一步验证 DGTO 算法的有效性, 对其与其

表5 各算法寻优结果对比

Table 5 Comparison of optimization results of each algorithm

函数	算法	最优值	最差值	平均值	标准差
f_1	DGTO	**0**	**0**	**0**	**0**
	GTO	0	0	0	0
	GWO	2.62×10^{-69}	6.96×10^{-63}	3.93×10^{-64}	1.28×10^{-63}
	DE	6.09×10^{-20}	1.86×10^{-18}	5.05×10^{-19}	4.44×10^{-19}
	WOA	2.04×10^{-94}	5.80×10^{-81}	1.23×10^{-82}	8.19×10^{-82}
f_2	DGTO	**0**	**0**	**0**	**0**
	GTO	7.39×10^{-217}	3.67×10^{-197}	7.69×10^{-199}	0
	GWO	3.25×10^{-39}	8.12×10^{-36}	5.90×10^{-37}	1.30×10^{-36}
	DE	1.91×10^{-12}	2.93×10^{-11}	9.68×10^{-12}	4.55×10^{-12}
	WOA	1.10×10^{-61}	5.26×10^{-52}	1.12×10^{-53}	7.43×10^{-53}
f_3	DGTO	**0**	**0**	**0**	**0**
	GTO	0	0	0	0
	GWO	3.82×10^{-37}	3.02×10^{-27}	8.81×10^{-29}	4.33×10^{-28}
	DE	2.63×10^{0}	2.74×10^{1}	8.17×10^{0}	5.41×10^{0}
	WOA	2.73×10^{-8}	6.89×10^{2}	1.16×10^{2}	1.46×10^{2}
f_4	DGTO	**0**	**0**	**0**	**0**
	GTO	2.01×10^{-218}	4.03×10^{-200}	1.18×10^{-201}	0
	GWO	9.71×10^{-24}	2.86×10^{-19}	1.79×10^{-20}	4.48×10^{-20}
	DE	2.60×10^{-5}	1.31×10^{-4}	6.72×10^{-5}	2.38×10^{-5}
	WOA	2.21×10^{-5}	2.20×10^{1}	1.35×10^{0}	4.37×10^{0}
f_5	DGTO	**7.46×10^{-30}**	**2.25×10^{-11}**	**6.34×10^{-13}**	**3.28×10^{-12}**
	GTO	4.91×10^{-15}	1.41×10^{0}	1.07×10^{-1}	3.06×10^{-1}
	GWO	5.32×10^{0}	9.54×10^{0}	6.70×10^{0}	7.69×10^{-1}
	DE	1.74×10^{0}	1.75×10^{1}	7.58×10^{0}	2.78×10^{0}
	WOA	3.49×10^{0}	8.95×10^{0}	6.58×10^{0}	6.38×10^{-1}
f_6	DGTO	**0**	**1.23×10^{-32}**	**5.55×10^{-34}**	**1.94×10^{-33}**
	GTO	9.24×10^{-32}	9.98×10^{-23}	2.18×10^{-24}	1.14×10^{-23}
	GWO	1.11×10^{-6}	6.33×10^{-6}	2.97×10^{-6}	1.13×10^{-6}
	DE	5.02×10^{-20}	2.93×10^{-18}	4.00×10^{-19}	4.42×10^{-19}
	WOA	2.62×10^{-5}	1.17×10^{-3}	3.12×10^{-4}	2.70×10^{-4}
f_7	DGTO	**-4.19×10^{3}**	**-4.19×10^{3}**	**-4.19×10^{3}**	**2.13×10^{-12}**
	GTO	-4.19×10^{3}	-4.19×10^{3}	-4.19×10^{3}	5.29×10^{-12}
	GWO	-3.56×10^{3}	-2.08×10^{3}	-2.84×10^{3}	3.65×10^{2}
	DE	-7.94×10^{7}	-8.23×10^{14}	-2.09×10^{13}	1.20×10^{14}
	WOA	-4.19×10^{3}	-2.49×10^{3}	-3.46×10^{3}	5.61×10^{2}
f_8	DGTO	**1.35×10^{-32}**	**1.35×10^{-32}**	**1.35×10^{-32}**	**1.11×10^{-47}**
	GTO	7.26×10^{-30}	7.46×10^{-2}	5.34×10^{-3}	1.25×10^{-2}
	GWO	1.15×10^{-6}	1.04×10^{-1}	1.00×10^{-2}	3.04×10^{-2}
	DE	4.37×10^{-21}	4.46×10^{-19}	8.94×10^{-20}	8.94×10^{-20}
	WOA	2.57×10^{-4}	1.10×10^{-1}	1.44×10^{-2}	2.57×10^{-2}
f_9	DGTO	**3.07×10^{-4}**	**3.07×10^{-4}**	**3.07×10^{-4}**	**1.74×10^{-19}**
	GTO	3.07×10^{-4}	1.22×10^{-3}	4.17×10^{-4}	3.01×10^{-4}
	GWO	3.07×10^{-4}	2.04×10^{-2}	2.78×10^{-3}	6.56×10^{-3}
	DE	5.16×10^{-4}	1.30×10^{-3}	8.97×10^{-4}	1.48×10^{-4}
	WOA	3.08×10^{-4}	2.17×10^{-3}	6.93×10^{-4}	4.89×10^{-4}
f_{10}	DGTO	**-3.32×10^{0}**	**-3.32×10^{0}**	**-3.32×10^{0}**	**0**
	GTO	-3.32×10^{0}	-3.20×10^{0}	-3.29×10^{0}	5.39×10^{-2}
	GWO	-3.32×10^{0}	-3.09×10^{0}	-3.27×10^{0}	7.57×10^{-2}
	DE	-3.32×10^{0}	-3.20×10^{0}	-3.31×10^{0}	3.17×10^{-2}
	WOA	-3.32×10^{0}	-2.43×10^{0}	-3.22×10^{0}	1.45×10^{-1}

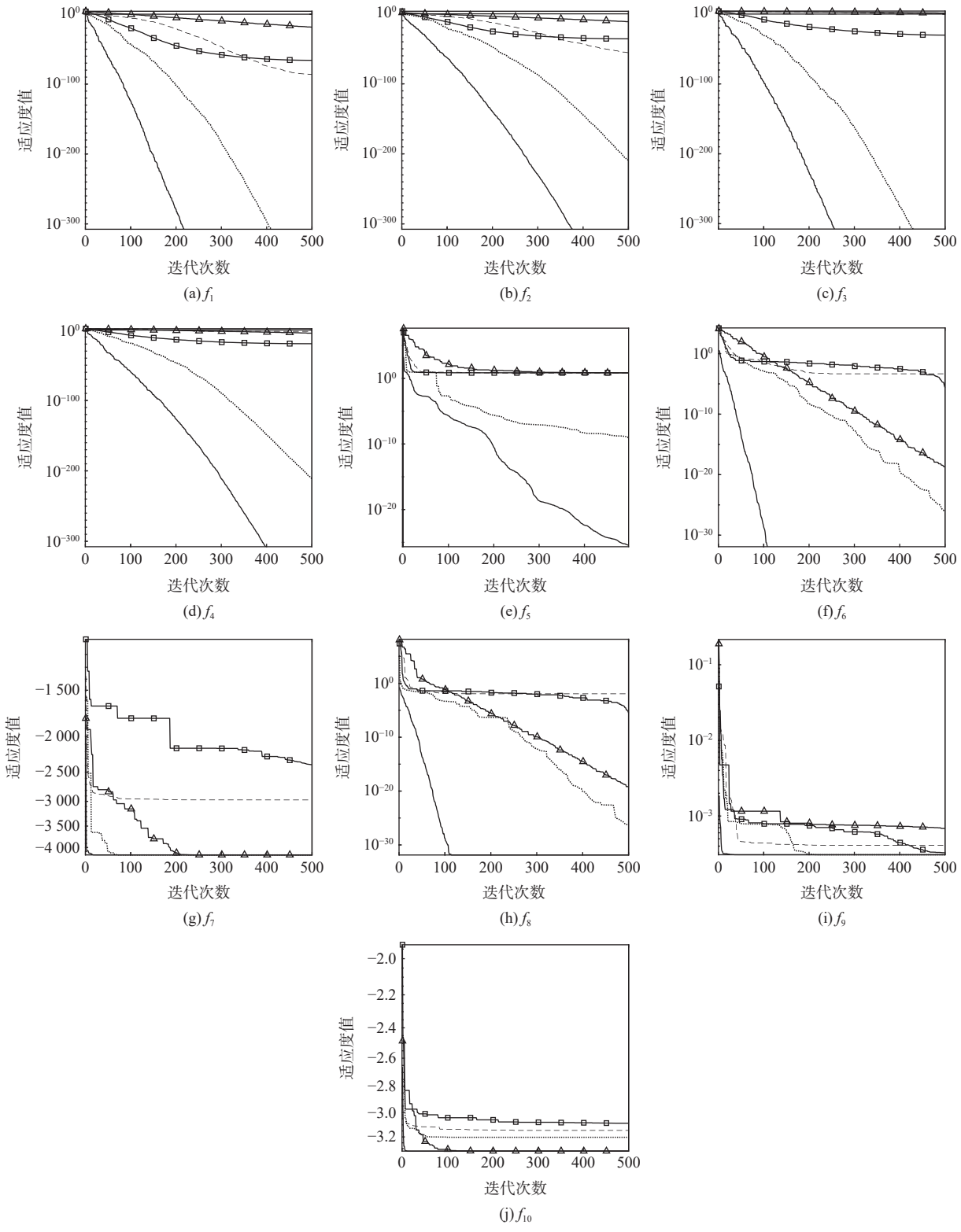


图 10　各算法在函数 f_1~f_{10} 的收敛曲线

Fig. 10　Convergence curves of each algorithm in functions f_1–f_{10}

他算法进行性能差异分析。对于 2 个独立的样本，常用的假设检验方法有参数检验和非参数检验。常见的参数假设检验方式有 t 检验和 z 检验，参数检验方法对数据要求较为严格，通常是在总体分布形式已知的情况下进行检验。对于非参数检验方式，符号检验和 Wilcoxon 秩和检验较为常见，这些方式在 2 个独立样本假设检验时无需知道样本的分布情况。而 Wilcoxon 秩和检验是在传统的符号检验方式上发展起来的，不仅可以直观显示 2 个独立样本是否有差异性，还可以通过 p 值判断出差异

性的程度，比传统的单独用正负号的检验更加有效。在对DGTO算法与其他算法进行性能差别分析前，样本分布情况未知，因此，在显著性水平为5%下，采用Wilcoxon秩和检验对DGTO算法与其余4个算法50次运行结果进行分析是否有显著的性能差异，并计算其p值。当p值小于5×10^{-2}时，否定对比算法与DGTO算法位置分布相似的假设[25]，即对比算法与DGTO算法的寻优性能具有明显差异。Wilcoxon秩和检验结果如表7所示。

表6 不同算法的平均绝对值误差

Table 6 Mean absolute error of different algorithms

算法	平均绝对值误差
DGTO	3.09×10^{-3}
GTO	1.73×10^{-2}
GWO	1.36×10^{2}
DE	2.09×10^{12}
WOA	8.54×10^{1}

由表7可知，DGTO算法与其他算法相比，大多数p值都小于5×10^{-2}，仅在函数f_1和f_3中，DGTO算法与GTO算法之间没有显著差异。因此，总体来说，与其他算法相比，DGTO算法的寻优性能被认为具有显著的优势。

表7 Wilcoxon秩和检验结果

Table 7 Wilcoxon rank sum test results

函数	p值			
	GTO	GWO	DE	WOA
f_1	NAN	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}
f_2	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}
f_3	NAN	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}
f_4	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}
f_5	7.97×10^{-18}	7.07×10^{-18}	7.07×10^{-18}	7.07×10^{-18}
f_6	6.35×10^{-19}	6.35×10^{-19}	6.35×10^{-19}	6.35×10^{-19}
f_7	4.32×10^{-12}	3.43×10^{-19}	3.43×10^{-19}	3.43×10^{-19}
f_8	1.30×10^{-18}	1.30×10^{-18}	1.30×10^{-18}	1.30×10^{-18}
f_9	2.79×10^{-16}	5.92×10^{-18}	5.92×10^{-18}	5.92×10^{-18}
f_{10}	2.71×10^{-7}	4.73×10^{-20}	6.69×10^{-11}	4.73×10^{-20}

注：参数p小于5×10^{-2}表示DGTO与对比算法有显著差异，NAN表示无法进行显著性判断。

4 工程优化设计问题

为验证DGTO算法改进的有效性，将其用于优化实际的拉伸/压缩弹簧设计问题。

4.1 拉伸/压缩弹簧设计问题

拉伸/压缩弹簧设计问题的目标是最小化弹簧拉伸/压缩的质量，其问题模型如图11所示。该优化设计问题受剪应力、振动频率和最小挠度约束条件的限制，包含3个决策变量，分别为弹簧线圈直径$d(x_1)$、平均线圈直径$D_a(x_2)$和有效绕圈数$P(x_3)$。

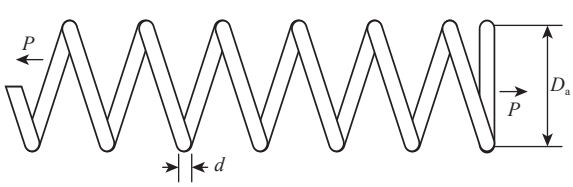


图11 弹簧设计模型

Fig. 11 Mode of spring design

拉伸/压缩弹簧设计问题的数学模型描述如下所示。

目标函数：

$$\min\ f(x)=(x_3+2)x_1^2x_2 \tag{23}$$

约束条件：

$$\begin{cases} g_1(x)=1-\dfrac{x_2^3x_3}{71\,785x_1^4}\leqslant 0\\ g_2(x)=\dfrac{4x_2^2-x_1x_2}{12\,566(x_1^3x_2-x_1^4)}+\dfrac{1}{5\,108x_1^2}-1\leqslant 0\\ g_3(x)=1-\dfrac{140.45x_1}{x_2^2x_3}\leqslant 0\\ g_4(x)=\dfrac{x_1+x_2}{1.5}-1\leqslant 0 \end{cases} \tag{24}$$

式中：$0.05\leqslant x_1\leqslant 2;0.25\leqslant x_2\leqslant 1.3;2\leqslant x_3\leqslant 15$。

4.2 实验结果及分析

为评估DGTO算法的性能，将其与标准GTO算法、GWO算法、DE算法、WOA算法、斑点鬣狗优化算法(spotted hyena optimizer, SHO)[26]、多元宇宙优化(multi-verse optimizer, MVO)算法[27]、正弦余弦算法(sine cosine algorithm, SCA)[28]和引力搜索算法(gravitational search algorithm, GSA)[29]进行实验对比，其中，SHO算法、MVO算法、SCA算法和GSA算法的实验数据来自文献[30]。各算法独立运行50次取最优值。各算法对拉伸/压缩弹簧设计问题的优化结果如表8所示。表中：加粗数据表示最优解。

表 8　弹簧设计问题各算法最优解

Table 8　Optimal solution of each algorithm for spring design problem

算法	d	D_a	P	最优解
DGTO	**0.051 72**	**0.357 50**	**11.242 94**	**0.012 665 252**
GTO	0.051 62	0.355 04	11.387 75	0.012 665 321
GWO	0.051 20	0.345 07	12.008 99	0.012 672 677
DE	0.051 61	0.354 79	11.403 14	0.012 665 876
WOA	0.051 48	0.351 82	11.582 31	0.012 666 117
SHO[30]	0.051 14	0.343 75	12.095 50	0.012 674 000
MVO[30]	0.050 00	0.315 96	14.226 23	0.012 816 930
SCA[30]	0.050 78	0.334 78	12.722 69	0.012 709 667
GSA[30]	0.050 00	0.317 31	14.228 67	0.012 873 881

由表 8 可知, DGTO 算法获得的使目标函数最小化的最优解为 $[x_1,x_2,x_3]$=[0.051 72,0.357 50,11.242 94], 最优值为 0.012 665 252。通过 DGTO 算法优化后的拉伸/压缩弹簧设计问题最优解优于其他算法最优解, 由此可说明 DGTO 算法在解决实际工程优化设计问题时具有一定的有效性, 进一步体现了算法的鲁棒性。

5　结　论

为改进标准 GTO 算法的寻优性能, 本文提出了一种基于双重随机扰动策略的人工大猩猩部队优化算法。

1) 在初始化阶段, 引入 Halton 序列对种群进行初始化, 增加了种群的多样性, 使得种群分布更加均匀。

2) 在探索阶段, 提出自适应位置搜索机制, 该机制有效提高了种群的收敛速度及收敛精度。

3) 为帮助算法跳出局部最优值, 提出双重随机扰动策略对种群位置进行扰动, 一重为柯西扰动, 另一重为中心位置扰动。

4) 在种群个体位置更新阶段, 采用逐维更新的方式对种群位置进行更新, 充分利用了种群个体每一维度的信息, 提高了算法的收敛精度。

5) 通过 10 个基准测试函数, 验证了本文算法改进的有效性, 并通过实际工程优化设计问题(即拉伸/压缩弹簧设计问题)证明了算法的鲁棒性, 为解决复杂的工程优化设计问题提供了新的思路。

在今后的工作中, 将会更加深入研究新的改进策略, 进一步提高标准 GTO 算法的性能。

参考文献（References）

[1] EBERHART R, KENNEDY J. A new optimizer using particle swarm theory[C]//Proceedings of the 6th International Symposium on Micro Machine and Human Science. Piscataway: IEEE Press, 2002: 39-43.

[2] CHEN D, ZHANG S, YANG Y, et al. Optimization of character image matching based on artificial bee colony algorithm[J]. Journal of Physics: Conference Series, 2021, 2035(1): 012034.

[3] MORIN M, ABI-ZEID I, QUIMPER C G. Ant colony optimization for path planning in search and rescue operations[J]. European Journal of Operational Research, 2023, 305(1): 53-63.

[4] AL-IBRAHIM A M H. Solving travelling salesman problem (TSP) by hybrid genetic algorithm (HGA)[J]. International Journal of Advanced Computer Science and Applications, 2020, 11(6): 376-384.

[5] LI J H, LEI Y S, YANG S H. Mid-long term load forecasting model based on support vector machine optimized by improved sparrow search algorithm[J]. Energy Reports, 2022, 8: 491-497.

[6] YAN Z P, ZHANG J Z, ZENG J, et al. Three-dimensional path planning for autonomous underwater vehicles based on a whale optimization algorithm[J]. Ocean Engineering, 2022, 250: 111070.

[7] GUHA D, ROY P K, BANERJEE S. Load frequency control of interconnected power system using grey wolf optimization[J]. Swarm and Evolutionary Computation, 2016, 27: 97-115.

[8] ABDOLLAHZADEH B, GHAREHCHOPOGH F S, MIRJALILI S. Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems[J]. International Journal of Intelligent Systems, 2021, 36(10): 5887-5958.

[9] XIAO Y N, SUN X, GUO Y L, et al. An improved gorilla troops optimizer based on lens opposition-based learning and adaptive β-hill climbing for global optimization[J]. Computer Modeling in Engineering & Sciences, 2022, 131(2): 815-850.

[10] LIANG Q W, CHU S C, YANG Q Y, et al. Multi-group gorilla troops optimizer with multi-strategies for 3D node localization of wireless sensor networks[J]. Sensors, 2022, 22(11): 4275.

[11] WU T Y, WU D, JIA H M, et al. A modified gorilla troops optimizer for global optimization problem[J]. Applied Sciences, 2022, 12(19): 10144.

[12] ALSOLAI H, ALZAHRANI J S, MARAY M, et al. Enhanced artificial gorilla troops optimizer based clustering protocol for UAV-assisted intelligent vehicular network[J]. Drones, 2022, 6(11): 358.

[13] MOSTAFA R R, GAHEEN M A, ABD ELAZIZ M, et al. An improved gorilla troops optimizer for global optimization problems and feature selection[J]. Knowledge-Based Systems, 2023, 269: 110462.

[14] BANGYAL W H, TAYYAB H, BATOOL H, et al. An improved particle swarm optimization algorithm with Chi-square mutation strategy[J]. International Journal of Advanced Computer Science and Applications, 2019, 10(3): 481-491.

[15] 宋立钦, 陈文杰, 陈伟海, 等. 基于混合策略的麻雀搜索算法改进及应用[J]. 北京航空航天大学学报, 2023, 49(8): 2187-2199.
SONG L Q, CHEN W J, CHEN W H, et al. Improvement and application of hybrid strategy-based sparrow search algorithm[J]. Journal of Beijing University of Aeronautics and Astronautics, 2023, 49(8): 2187-2199(in Chinese).

[16] 周理, 朱红求. 基于自适应步长果蝇算法的爬行机器人足端轨迹规划[J]. 机械设计与研究, 2021, 37(3): 60-63.
ZHOU L, ZHU H Q. Foot trajectory planning of creeping robot based on adaptive step fruit fly optimization algorithm[J]. Machine Design & Research, 2021, 37(3): 60-63(in Chinese).

[17] 宋阿妮, 包贤哲, 权轶. 基于混沌自适应萤火虫算法的 UAVs 分配策略[J]. 计算机应用与软件, 2022, 39(2): 300-306.
SONG A N, BAO X Z, QUAN Y. Uavs scheduling strategy based on chaotic adaptive firefly algorithm[J]. Computer Applications and

Software, 2022, 39(2): 300-306(in Chinese).

[18] 李凡长, 刘洋, 吴鹏翔, 等. 元学习研究综述[J]. 计算机学报, 2021, 44(2): 422-446.
LI F C, LIU Y, WU P X, et al. A survey on recent advances in meta-learning[J]. Chinese Journal of Computers, 2021, 44(2): 422-446 (in Chinese).

[19] LI K W, LI S H, HUANG Z C, et al. Grey wolf optimization algorithm based on Cauchy-Gaussian mutation and improved search strategy[J]. Scientific Reports, 2022, 12: 18961.

[20] 刘薇, 赵剑锟, 刘义保, 等. 基于改进型灰狼算法的γ能谱解析应用研究[J]. 核技术, 2021, 44(4): 31-36.
LIU W, ZHAO J K, LIU Y B, et al. Application research of γ energy spectrum analysis based on improved grey wolf algorithm[J]. Nuclear Techniques, 2021, 44(4): 31-36(in Chinese).

[21] 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.

[22] 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.

[23] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[24] NABIL E. A modified flower pollination algorithm for global optimization[J]. Expert Systems with Applications, 2016, 57: 192-203.

[25] 张新明, 王霞, 康强. 改进的灰狼优化算法及其高维函数和FCM优化[J]. 控制与决策, 2019, 34(10): 2073-2084.
ZHANG X M, WANG X, KANG Q. Improved grey wolf optimizer and its application to high-dimensional function and FCM optimization[J]. Control and Decision, 2019, 34(10): 2073-2084(in Chinese).

[26] LUO Q F, LI J, ZHOU Y Q, et al. Using spotted hyena optimizer for training feedforward neural networks[J]. Cognitive Systems Research, 2021, 65: 1-16.

[27] ABUALIGAH L. Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications[J]. Neural Computing and Applications, 2020, 32(16): 12381-12401.

[28] MIRJALILI S. SCA: a sine cosine algorithm for solving optimization problems[J]. Knowledge-Based Systems, 2016, 96: 120-133.

[29] YAZDANI S, NEZAMABADI-POUR H, KAMYAB S. A gravitational search algorithm for multimodal optimization[J]. Swarm and Evolutionary Computation, 2014, 14: 1-14.

[30] KAUR S, AWASTHI L K, SANGAL A L, et al. Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization[J]. Engineering Applications of Artificial Intelligence, 2020, 90: 103541.

Artificial gorilla troops optimizer based on double random disturbance and its application of engineering problem

DU Xiaoxin[1, 2, *], HAO Tianru[1], WANG Bo[1, 2], WANG Zhenfei[1], ZHANG Jianfei[1, 2], JIN Mei[1, 2]

(1. College of Computer and Control Engineering, Qiqihar University, Qiqihar 161006, China;
2. Heilongjiang Key Laboratory of Big Data Network Security Detection and Analysis, Qiqihar University, Qiqihar 161006, China)

Abstract: Traditional artificial gorilla troops optimizer (GTO) has the drawbacks of easily falling into local optimum, slow convergence speed, and low optimization accuracy. Aiming at these problems, an artificial gorilla troops optimizer based on a double random disturbance strategy (DGTO) was proposed. Firstly, the Halton sequence was introduced to initialize the population to increase the diversity of the population. Secondly, the method's convergence speed was increased by using the multi-dimensional random number technique during the algorithm optimization stage and proposing an adaptive position exploration mechanism. Thirdly, a double random disturbance strategy was proposed, which solved the group effect of gorillas and enhanced the ability of the algorithm to jump out of the local optimum. Finally, the individual position was updated by a dimension-by-dimension update strategy, which improved the convergence accuracy of the algorithm. It is evident that the enhanced technique has a greater improvement in optimization accuracy and convergence speed when comparing the Wilcoxon rank sum test results with the optimization results of ten benchmark test functions. In addition, through the experimental comparative analysis of one practical engineering optimization problem, the superiority of the proposed algorithm in dealing with practical engineering problems is further verified.

Keywords: artificial gorilla troops optimizer; Halton sequence; adaptive position exploration; double random disturbance strategy; dimension-by-dimension update

Received: 2023-06-25; **Accepted**: 2023-09-11; **Published Online**: 2023-10-20 11:49

URL: link.cnki.net/urlid/11.2625.V.20231020.0903.001

Foundation item: Heilongjiang Provincial Higher Education Institutions Basic Scientific Research Business Funds Natural Science Young Innovative Talents Program (145209206)

* **Corresponding author**. E-mail: xiaoxindu@qqhru.edu.cn