doi：10.13413/j.cnki.jdxblxb.2023231

适应度步长的全局局部协同优化算法

初雅莉[1]，韩旭明[2,3]，王晏泽[2]，吕 帅[2]
（1. 长春工业大学 数学与统计学院，长春 130012；2. 暨南大学 信息科学技术学院，广州 510632；
3. 可信人工智能教育部工程研究中心，广州 510632）

摘要：针对现有优化算法求解精度低的问题，提出一种适应度步长的全局局部协同优化算法. 该算法通过均衡化个体适应度，动态分配每次迭代中个体的全局搜索步长和局部搜索步长，实现了算法在解空间内全局搜索和局部搜索的有效协同，进而提升了求解精度. 实验结果表明，该算法在基准函数测试中展现了较高的精度和良好的稳定性，并通过仿真实验验证了其在解决复杂工程优化问题中的有效性.

关键词：优化算法；均衡化适应度值；适应度步长；协同搜索；工程优化
中图分类号：TP301 **文献标志码**：A **文章编号**：1671-5489(2024)06-1419-07

Global-Local Cooperative Optimization Algorithm with Fitness Step Size

CHU Yali[1], HAN Xuming[2,3], WANG Yanze[2], LÜ Shuai[2]
(1. *School of Mathematics and Statistics, Changchun University of Technology, Changchun* 130012, *China*;
2. *College of Information Science and Technology, Jinan University, Guangzhou* 510632, *China*;
3. *Engineering Research Center of Trustworthy AI of Ministry of Education, Guangzhou* 510632, *China*)

Abstract: Aiming at the problem of low solution precision in existing optimization algorithms, we proposed a global-local cooperative optimization algorithm with fitness step size. The algorithm achieved effective collaboration between global and local search in the solution space by balancing individual fitness and dynamically allocating global and local search step sizes during each iteration, thereby enhancing the solution precision. Experimental results show that the proposed algorithm has high precision and stability in benchmark function tests, and its effectiveness in solving complex engineering optimization problems is verified through simulation experiments.

Keywords: optimization algorithm; normalized fitness value; fitness step size; cooperative search; engineering optimization

优化问题因其变量数量多且约束条件复杂，导致计算复杂度较高，搜索空间也随之变得极其复杂[1]，制约了传统优化算法在寻找全局最优解方面的效率和准确性. 群体智能优化算法具有较强的鲁棒性和通用性，在实际应用中优势显著[2-4]. 目前，已构建了基于个体间信息共享和合作机制的群体智能优化算法，在一定程度上缓解了优化算法易陷入局部最优解的问题[5-7]. 但现有部分群体智能优化算法的求解性能仍不足，主要体现在探索广阔搜索空间时算法难以保持足够的多样性，种群中个体搜

收稿日期：2023-06-06.

第一作者简介：初雅莉(1992—)，女，汉族，硕士研究生，从事智能计算和大数据分析的研究，E-mail：chuyl-1992@163.com.
通信作者简介：韩旭明(1971—)，男，汉族，博士，教授，从事进化计算和机器学习的研究，E-mail：hanxuming@jnu.edu.cn.

基金项目：国家社会科学基金(批准号：22BTJ057；24BTJ041；24CTJ034)和吉林省自然科学基金(批准号：20200201164JC).

索步长难以自适应调节，进而导致在逼近最优解时全局搜索与局部搜索能力失衡.

为克服上述问题，本文提出一种适应度步长的全局局部协同优化(GLCOFS)算法. 该算法通过均衡化适应度值，以自适应动态设置个体在解空间中的搜索步长，即适应度步长，有效协同全局搜索和局部搜索，平衡算法的勘探和开发能力. 在全局搜索中，GLCOFS算法利用全局适应度步长指导个体围绕当前全局最优解进行精细搜索，提升算法的开发能力以逼近更优解；在局部搜索中，个体以其当前位置为基点，结合局部适应度步长以及全局最优个体和随机个体信息，拓宽搜索范围，挖掘潜在优秀解，增强算法的勘探能力. GLCOFS算法在每次迭代中灵活调配个体的搜索行为，实现全局与局部搜索的紧密协同，有助于种群跳出局部极值，提升优化问题解的精度.

1 GLCOFS 算法

1.1 种群初始化

GLCOFS算法初始化种群，先生成一个均匀分布的随机群体 $X=\{X_1,\cdots,X_i,\cdots,X_N\}$，其中 N 表示种群中个体数量，$X_i=\{X_{i,1},\cdots,X_{i,d},\cdots,X_{i,\dim}\}$表示第 i 个个体，dim 为优化问题的维度. 然后用下式获得初始群体：

$$X_{i,d}=X_{\min}+r_1(X_{\max}-X_{\min}), \tag{1}$$

其中 $X_{i,d}$为种群中第 i 个个体在第 d 维搜索空间的位置，r_1 为 0～1 内的随机数，$X_{\min}$和 $X_{\max}$分别为优化问题的下限和上限.

1.2 种群动态划分

GLCOFS算法在每次迭代中将种群动态随机划分为两组：$X^{t,G1}$和 $X^{t,G2}$，分别侧重于全局搜索和局部搜索. 第 t 时刻种群划分如下：

$$\begin{cases} X^{t,G1}=\mathrm{randSelect}(X^t,N^{t,G1}), \\ X^{t,G2}=X^t-X^{t,G1}, \end{cases} \tag{2}$$

其中：X^t 表示 t 时刻的种群；randSelect(·)表示从种群中随机选择出 $N^{t,G1}$个体作为 $X^{t,G1}$中成员，剩余个体加入第二组 $X^{t,G2}$. 第 t 时刻 $X^{t,G1}$和 $X^{t,G2}$中个体数量为

$$\begin{cases} N^{t,G1}=\lfloor r_2\times N\rfloor, \\ N^{t,G2}=N-N^{t,G1}, \end{cases} \tag{3}$$

其中 r_2 为[0,1]内的一个随机数，用于控制每次迭代中第一组个体数量；N 为种群中个体总量. $\lfloor\cdot\rfloor$为获取不大于符号内实数的最大整数.

1.3 适应度步长的全局局部协同搜索

1) 均衡化适应度. 其目的是通过消除每次迭代中个体适应度之间的量纲差异，使个体能自适应调整搜索步长，从而有效指导个体进行多样化搜索. 均衡化是将每次迭代中种群内所有个体的适应度值映射至[0,1]内，计算公式为

$$F_{\mathrm{normStep}}(X_i^t)=\frac{F(X_i^t)-F_{\min}}{F_{\max}-F_{\min}}, \tag{4}$$

其中 $F_{\max}$和 $F_{\min}$分别为 t 时刻种群中的最大适应度值和最小适应度值；$F(X_i^t)$表示种群中 t 时刻第 i 个个体的适应度值. 均衡化适应度融入全局搜索与局部搜索中，以动态调整不同个体的移动步长并引导搜索行为.

2) 适应度步长引导全局搜索. 第一组个体 $X^{t,G1}$利用其在当前时刻的适应度步长，在种群全局最优解邻域进行全局搜索. 第一组中第 i 个个体 $X_i^{t,G1}$ 的全局适应度步长计算公式为

$$\mu_{\mathrm{gStep}}=\cos\left(F_{\mathrm{normStep}}(X_i^{t,G1})\times r_3\times\pi\times\frac{t}{T}\right), \tag{5}$$

其中 r_3 为[0,1]内的一个随机数，T 表示最大迭代次数. 在全局搜索中，通过 μ_{gStep}确定每个个体在下一时刻的搜索步长. 对于远离全局最优解的个体，μ_{gStep}值较大，个体以较大步长搜索解空间；而对于接近全局最优解的个体，则减小步长进行精细搜索，计算公式为

$$X_{i,d}^{t+1,G1}=\begin{cases}X_d^{t,\text{best}}+\mu_{\text{gStep}}(X_d^{t,\text{best}}-X_{i,d}^{t,G1}), & r_4<0.5,\\ X_d^{t,\text{best}}+\mu_{\text{gStep}}S_{\text{rand}}, & \text{其他},\end{cases}\tag{6}$$

其中 $X_d^{t,\text{best}}$ 为种群在第 t 时刻的全局最优解，S_{rand} 为搜索空间中随机位置，r_4 为[0,1]内的一个随机数.

3）适应度步长引导局部搜索. 第二组个体 $X^{t,G2}$ 旨在结合自身适应度步长，在解空间进行局部搜索，其局部适应度步长计算公式为

$$\mu_{\text{lStep}}=F_{\text{normStep}}(X_j^{t,G2})\times\frac{t}{T}.\tag{7}$$

第二组个体 $X^{t,G2}$ 通过与全局最优个体和随机个体进行信息共享，并利用其适应度步长，在当前个体领域展开搜索，计算公式为

$$X_{j,d}^{t+1,G2}=\begin{cases}X_{j,d}^{t,G2}+\mu_{\text{lStep}}(X_d^{t,\text{best}}-X_{j,d}^{t,G2}), & r_5<0.5,\\ X_{j,d}^{t,G2}+|r_n|(X_d^{t,\text{rand}}-X_{j,d}^{t,G2}), & \text{其他},\end{cases}\tag{8}$$

其中 $X_d^{t,\text{best}}$ 和 $X_d^{t,\text{rand}}$ 分别表示全局最优解和随机个体在 t 时刻 d 维中的位置，r_n 为服从正态分布的随机数，r_5 为服从[0,1]内均匀分布的随机数.

1.4　计算时间复杂度分析及算法流程

GLCOFS 算法的复杂度取决于种群数量 N、最大迭代次数 T 和问题的维度 dim. GLCOFS 算法的主循环执行 T 次迭代，每次迭代中，种群中 N 个个体分为两组：第一组个体 $X^{t,G1}$ 采用全局搜索解空间，第二组个体 $X^{t,G2}$ 对解空间进行局部搜索，其算法流程如图 1 所示. 因此，GLCOFS 算法的时间复杂度为

$$O(\text{GLCOFS})=O(T(O(\text{全局搜索})+O(\text{局部搜索})))=O(T(N^{G1}\dim+N^{G2}\dim))=O(TN\dim).\tag{9}$$

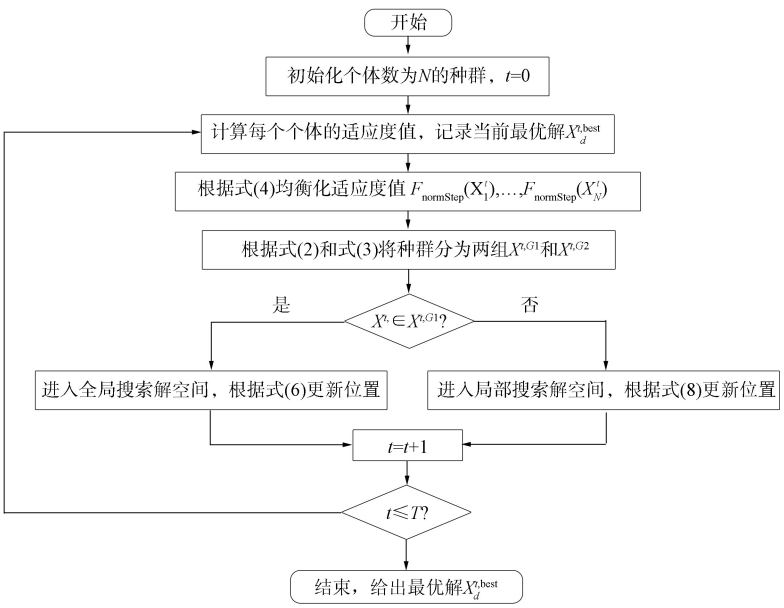

图 1　GLCOFS 算法流程

Fig. 1　Flow chart of GLCOFS algorithm

2　实验及结果分析

2.1　实验配置

1）基准函数：选取如表 1 所示的 CEC2017[8] 中的 15 个复杂函数，并设定搜索空间维度为 100 进行测试，以评估并验证 GLCOFS 算法在解决优化问题上的性能与效果.

2）对比算法：将 GLCOFS 算法与 4 种最新的群智能优化算法 HO,GKSO,COA,MGO 进行比较.

3）参数设置：所有算法的种群个体数量为 30，最大评估次数为 30 000，以确保实验的可比较性. 对每种算法独立执行 30 次，以减少随机因素对测试结果的影响.

4) 实验环境：所有对比实验均在64位版本的Windows 10操作系统下使用MATLAB R2020a软件进行实验. 硬件平台配置为运行频率为3.20 GHz的英特尔酷睿i7-8700处理器，8 GB内存.

表1 基准函数信息

Table 1 Benchmark function information

函数	函数名称	最优值	函数	函数名称	最优值
F_1	Shifted and rotated Rosenbrock's function	400	F_9	Hybrid function 6 ($N=4$)	1 600
F_2	Shifted and rotated Rastrigin's function	500	F_{10}	Hybrid function 6 ($N=5$)	1 700
F_3	Shifted and rotated expanded Scaffer's F_6 function	600	F_{11}	Hybrid function 6 ($N=5$)	1 800
F_4	Shifted and rotated Lunacek Bi-Rastrigin function	700	F_{12}	Composition function 7 ($N=6$)	2 700
F_5	Shifted and rotated non-continuous Rastrigin's function	800	F_{13}	Composition function 8 ($N=6$)	2 800
F_6	Shifted and rotated Lévy function	900	F_{14}	Composition function 9 ($N=3$)	2 900
F_7	Hybrid function 4 ($N=4$)	1 400	F_{15}	Composition function 10 ($N=3$)	3 000
F_8	Hybrid function 5 ($N=4$)	1 500			

2.2 优化结果与分析

在100维基准函数上，比较GLCOFS算法与4种先进优化算法的性能，对比适应度值的平均值与标准差值，并给出每种算法根据适应度值平均值的排名，结果列于表2. 由表2可见，GLCOFS算法达到了最高的平均求解精度，而其平均适应度值在所有对比算法中均位列第一，在14个函数上，GLCOFS算法的适应度标准差最小，表明了GLCOFS算法在解决复杂优化问题上的有效性.

表2 不同算法对基准函数(dim=100)优化结果的比较

Table 2 Comparison of optimization results for benchmark functions (dim=100) by different algorithms

函数	指标	GLCOFS算法	HO算法	GKSO算法	COA算法	MGO算法
F_1	平均值	$9.601\,374\times10^{2}$	$1.260\,200\times10^{4}$	$6.155\,695\times10^{3}$	$1.105\,367\times10^{5}$	$1.460\,026\times10^{3}$
	标准差	$8.397\,480\times10^{1}$	$1.681\,980\times10^{3}$	$1.260\,722\times10^{3}$	$1.194\,395\times10^{4}$	$1.854\,704\times10^{2}$
	排名	1	4	3	5	2
F_2	平均值	$8.156\,010\times10^{2}$	$1.615\,716\times10^{3}$	$1.772\,236\times10^{3}$	$2.135\,474\times10^{3}$	$1.405\,925\times10^{3}$
	标准差	$4.165\,232\times10^{1}$	$5.405\,559\times10^{1}$	$7.614\,188\times10^{1}$	$2.828\,819\times10^{1}$	$7.319\,337\times10^{1}$
	排名	1	3	4	5	2
F_3	平均值	$6.130\,268\times10^{2}$	$6.861\,284\times10^{2}$	$6.936\,743\times10^{2}$	$7.129\,062\times10^{2}$	$6.695\,324\times10^{2}$
	标准差	$2.372\,771\times10^{0}$	$3.731\,012\times10^{0}$	$6.486\,470\times10^{0}$	$3.579\,195\times10^{0}$	$1.104\,902\times10^{1}$
	排名	1	3	4	5	2
F_4	平均值	$1.235\,310\times10^{3}$	$3.493\,885\times10^{3}$	$3.170\,166\times10^{3}$	$3.999\,680\times10^{3}$	$2.696\,566\times10^{3}$
	标准差	$5.305\,133\times10^{1}$	$1.080\,660\times10^{2}$	$2.260\,080\times10^{2}$	$8.305\,055\times10^{1}$	$2.096\,175\times10^{2}$
	排名	1	4	3	5	2
F_5	平均值	$1.137\,944\times10^{3}$	$2.072\,931\times10^{3}$	$2.207\,225\times10^{3}$	$2.598\,606\times10^{3}$	$1.754\,975\times10^{3}$
	标准差	$4.989\,132\times10^{1}$	$5.844\,060\times10^{1}$	$8.121\,701\times10^{1}$	$5.815\,975\times10^{1}$	$9.240\,210\times10^{1}$
	排名	1	3	4	5	2
F_6	平均值	$5.900\,125\times10^{3}$	$5.221\,664\times10^{4}$	$7.109\,411\times10^{4}$	$8.024\,308\times10^{4}$	$6.079\,275\times10^{4}$
	标准差	$1.262\,303\times10^{3}$	$3.678\,465\times10^{3}$	$4.898\,607\times10^{3}$	$4.267\,532\times10^{3}$	$6.815\,864\times10^{3}$
	排名	1	2	4	5	3
F_7	平均值	$1.110\,915\times10^{6}$	$1.126\,328\times10^{7}$	$5.600\,508\times10^{6}$	$1.138\,132\times10^{8}$	$2.943\,394\times10^{6}$
	标准差	$8.996\,349\times10^{5}$	$3.602\,800\times10^{6}$	$2.544\,927\times10^{6}$	$4.796\,396\times10^{7}$	$1.024\,955\times10^{6}$
	排名	1	4	3	5	2
F_8	平均值	$2.669\,540\times10^{3}$	$2.594\,761\times10^{7}$	$3.915\,582\times10^{7}$	$2.582\,807\times10^{10}$	$3.651\,774\times10^{5}$
	标准差	$7.974\,965\times10^{2}$	$4.902\,684\times10^{7}$	$3.027\,271\times10^{7}$	$4.434\,751\times10^{9}$	$5.497\,000\times10^{5}$
	排名	1	3	4	5	2
F_9	平均值	$5.331\,748\times10^{3}$	$1.190\,554\times10^{4}$	$1.079\,959\times10^{4}$	$2.528\,697\times10^{4}$	$7.139\,954\times10^{3}$
	标准差	$5.370\,098\times10^{2}$	$1.373\,015\times10^{3}$	$1.100\,304\times10^{3}$	$2.677\,355\times10^{3}$	$8.273\,886\times10^{2}$
	排名	1	4	3	5	2

续表 2

Continued to table 2

函数	指标	GLCOFS 算法	HO 算法	GKSO 算法	COA 算法	MGO 算法
F_{10}	平均值	$4.603\,308\times10^{3}$	$8.873\,698\times10^{3}$	$7.196\,131\times10^{3}$	$1.412\,573\times10^{7}$	$5.718\,099\times10^{3}$
	标准差	$4.995\,899\times10^{2}$	$1.610\,039\times10^{3}$	$8.074\,830\times10^{2}$	$1.140\,230\times10^{7}$	$7.641\,342\times10^{2}$
	排名	1	4	3	5	2
F_{11}	平均值	$1.978\,153\times10^{6}$	$9.414\,152\times10^{6}$	$6.670\,109\times10^{6}$	$3.209\,736\times10^{8}$	$4.112\,579\times10^{6}$
	标准差	$9.992\,850\times10^{5}$	$5.295\,752\times10^{6}$	$3.474\,656\times10^{6}$	$1.406\,544\times10^{8}$	$2.200\,170\times10^{6}$
	排名	1	4	3	5	2
F_{12}	平均值	$3.775\,492\times10^{3}$	$6.016\,226\times10^{3}$	$4.677\,017\times10^{3}$	$1.598\,478\times10^{4}$	$4.027\,230\times10^{3}$
	标准差	$5.888\,775\times10^{1}$	$6.381\,709\times10^{2}$	$3.572\,962\times10^{2}$	$1.776\,661\times10^{3}$	$1.982\,505\times10^{2}$
	排名	1	4	3	5	2
F_{13}	平均值	$3.947\,218\times10^{3}$	$1.194\,986\times10^{4}$	$8.246\,011\times10^{3}$	$3.075\,194\times10^{4}$	$4.523\,767\times10^{3}$
	标准差	$1.262\,172\times10^{2}$	$1.240\,042\times10^{3}$	$7.392\,694\times10^{2}$	$1.413\,620\times10^{3}$	$2.610\,660\times10^{2}$
	排名	1	4	3	5	2
F_{14}	平均值	$6.642\,598\times10^{3}$	$1.626\,382\times10^{4}$	$1.322\,172\times10^{4}$	$6.162\,071\times10^{5}$	$9.355\,215\times10^{3}$
	标准差	$5.346\,998\times10^{2}$	$2.516\,344\times10^{3}$	$1.260\,196\times10^{3}$	$3.446\,339\times10^{5}$	$7.342\,580\times10^{2}$
	排名	1	4	3	5	2
F_{15}	平均值	$2.525\,482\times10^{5}$	$1.554\,605\times10^{9}$	$7.611\,760\times10^{8}$	$4.313\,745\times10^{10}$	$3.355\,670\times10^{7}$
	标准差	$1.197\,926\times10^{5}$	$5.056\,477\times10^{8}$	$3.513\,377\times10^{8}$	$6.502\,127\times10^{9}$	$1.307\,558\times10^{7}$
	排名	1	4	3	5	2

图 2 为 GLCOFS 算法与对比算法在部分基准函数上的收敛曲线．由图 2 可见，GLCOFS 算法在基准函数上的收敛性能均优于其他对比算法．这一优势主要归因于 GLCOFS 算法采用了个体均衡化适应度值以协同全局搜索和局部搜索，使得在处理包含多个局部最优解的复杂优化问题时，GLCOFS 算法展现了较强的搜索能力，有效平衡了算法的探索和开发能力．

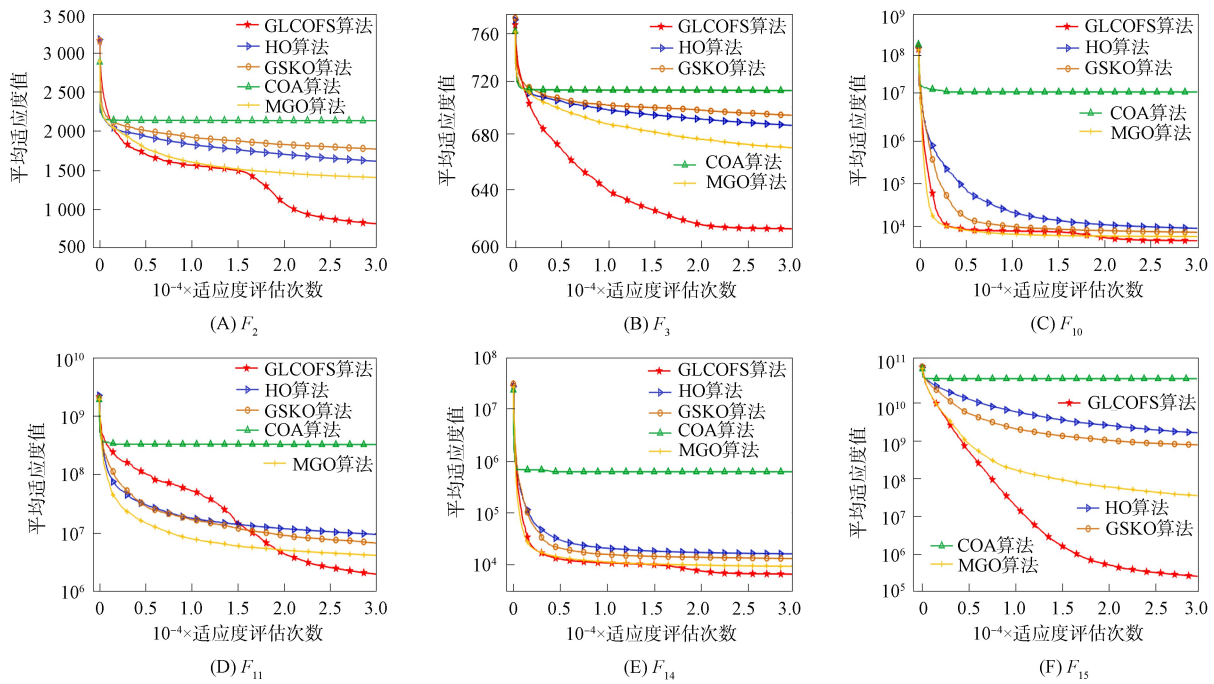

图 2　GLCOFS 算法与对比算法在部分基准函数上的收敛曲线

Fig. 2　Convergence curves of GLCOFS algorithm and comparative algorithms on partial benchmark functions

下面给出所有算法在基准函数上运行 30 次后获得的最佳适应度值的分布，以分析各算法的稳定性．图 3 为 GLCOFS 算法与对比算法在部分基准函数上的箱型图．由图 3 可见，GLCOFS 算法的箱型图面积相对较小，最佳适应度值位于较低水平，表明 GLCOFS 算法在多次运行中能产生更集中的解．

GLCOFS算法在上边缘外异常值(图中“+”处)的频率远低于其他算法，进一步表明GLCOFS算法能在复杂优化问题下稳定地找到近似最优解.

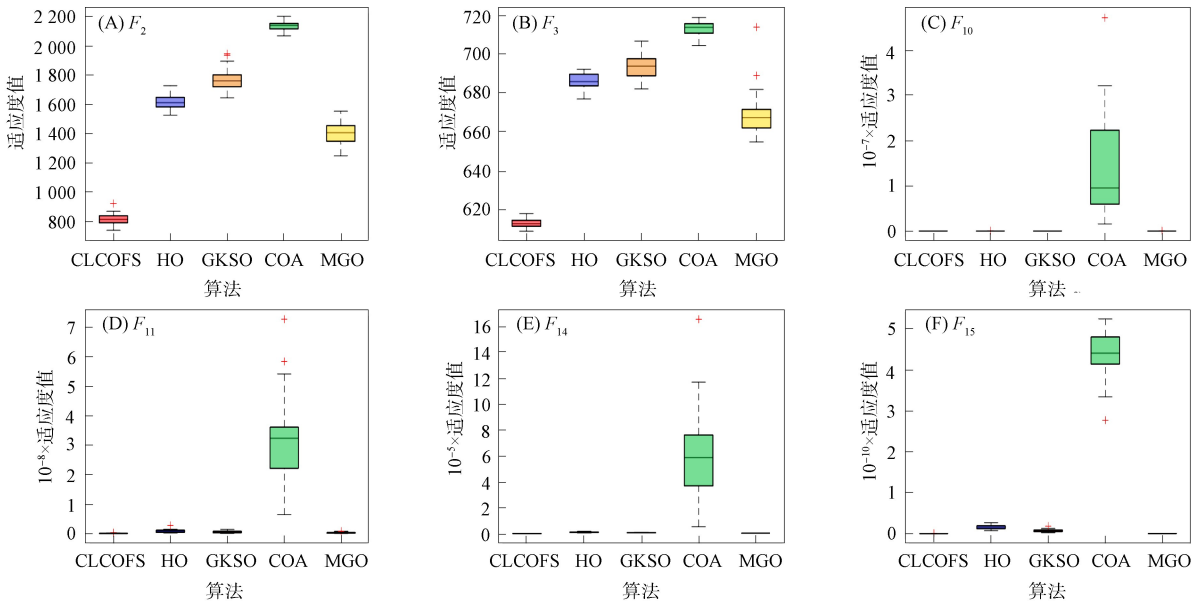

图3 GLCOFS算法与对比算法在部分基准函数上的箱型图

Fig. 3 Box plot of GLCOFS algorithm and comparative algorithms on partial benchmark functions

3 GLCOFS算法应用

工程设计问题因其具有复杂的目标函数和大量约束条件而很难求解. 为进一步验证GLCOFS算法的实用性和有效性，下面在有代表性的高约束工程优化问题——压力容器设计中进行实验对比. 压力容器设计的主要目标是以最低的成本制造出一个合格的压力容器[9]. 该问题需要确定以下4个设计变量：壳体厚度 x_1、封头厚度 x_2、内半径 x_3 和忽略头部的圆柱截面长度 x_4. 压力容器设计的数学模型为

$$\begin{aligned}&\min\ f(\boldsymbol{X})=0.622\,4x_1x_3x_4+1.778\,1x_2x_3^2+3.166\,1x_1^2x_4+19.84x_1^2x_3,\\&\text{s.t.}\ \ g_1(\boldsymbol{X})=-x_1+0.019\,3x_3\leqslant 0,\\&\qquad g_2(\boldsymbol{X})=-x_2+0.009\,54x_3\leqslant 0,\\&\qquad g_3(\boldsymbol{X})=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3+1\,296\,000\leqslant 0,\\&\qquad g_4(\boldsymbol{X})=x_4-240\leqslant 0.\end{aligned}\tag{10}$$

在对比实验中，各算法独立求解同一问题30次，通过收集优化结果的最优值、平均值、标准差和中位数全面评估GLCOFS算法. 实验参数与基准函数实验保持一致.

表3列出了不同算法解决压力容器设计问题的对比结果.

表3 不同算法解决压力容器设计问题的对比结果

Table 3 Comparison results of different algorithms for solving pressure vessel design problems

算法	优化结果				变量最优值			
	最优值	平均值	标准差	中位数	x_1	x_2	x_3	x_4
GLCOFS	5.885×10^3	6.346×10^3	2.643×10^2	6.881×10^3	0.778 200	0.384 664	40.321 223	199.977 669
HO	6.138×10^3	6.785×10^3	3.599×10^2	7.436×10^3	0.898 797	0.446 081	46.528 721	128.565 497
GKSO	6.063×10^3	6.923×10^3	4.390×10^2	7.711×10^3	0.822 526	0.405 284	41.873 777	179.724 082
COA	1.166×10^4	6.842×10^4	6.053×10^4	2.087×10^5	1.261 409	1.270 854	56.132 394	56.410 659
MGO	5.886×10^3	6.351×10^3	5.191×10^2	7.319×10^3	0.778 210	0.384 722	40.320 593	199.999 914

由表3可见，GLCOFS算法可有效求解压力容器设计问题，且有较好的收敛性，如图4所示.

GLCOFS 算法获得了低于其他算法的最优值，展示了其出色的优化能力，且其标准差较低，揭示了算法运行结果的高度集中性，即多次求解间波动小. 此外，GLCOFS 算法的中位数较低，进一步验证了算法的稳定性和高效性.

综上所述，针对现有优化算法求解精度低的问题，本文提出了一种适应度步长的全局局部协同优化(GLCOFS)算法. 该算法基于均衡化适应度值动态调整全局与局部搜索步长，实现了全局搜索与局部搜索的有效协同；通过协同搜索机制，有效缓解了优化问题中算法易陷入局部最优解的问题，显著提高了求解问题的精度. 实验结果表明，在多组复杂的多峰、混合和组合型基准函数上，GLCOFS 算法在收敛精度和稳定性方面均优于其他最新优化算法. 在处理含有多个局部最优解的复杂问题时，GLCOFS 算法能稳定地找到全局近似最优解，有实际工程应用价值.

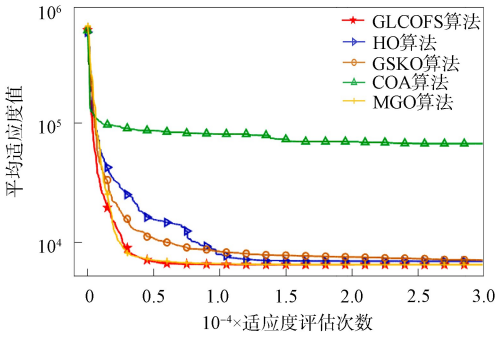

图 4　压力容器设计问题的收敛曲线

Fig. 4　Convergence curves of pressure vessel design problems

参　考　文　献

[1] 刘晴，王天昊，徐旭. 深度学习神经网络的新型自适应激活函数 [J]. 吉林大学学报(理学版)，2019，57(4)：857-859. (LIU Q，WANG T H，XU X. New Adaptive Activation Function for Deep Learning Neural Networks [J]. Journal of Jilin University (Science Edition)，2019，57(4)：857-859.)

[2] 陈少淼，陈瑞，梁伟，等. 面向复杂约束优化问题的进化算法综述 [J]. 软件学报，2022，34(2)：565-581. (CHEN S M，CHEN R，LIANG W，et al. Overview of Evolutionary Algorithms for Complex Constrained Optimization Problems [J]. Journal of Software，2022，34(2)：565-581.)

[3] 刘威，高新成，王启龙，等. 基于离散粒子群的 SDN 动态流调度算法 [J]. 吉林大学学报 (理学版)，2023，61(5)：1592-1600. (LIU W，GAO X C，WANG Q L，et al. SDN Dynamic Flow Scheduling Algorithm Based on Discrete Particle Swarm Optimization [J]. Journal of Jilin University (Science Edition)，2023，61(5)：1592-1600.)

[4] HU G，ZHONG J Y，WEI G，et al. DTCSMO：An Efficient Hybrid Starling Murmuration Optimizer for Engineering Applications [J]. Computer Methods in Applied Mechanics and Engineering，2023，405：115878-1-115878-79.

[5] AMIRI M H，MEHRABI H N，MONTAZERI M，et al. Hippopotamus Optimization Algorithm：A Novel Nature-Inspired Optimization Algorithm [J]. Scientific Reports，2024，14(1)：5032-1-5032-14.

[6] MOHAMMED H，RASHID T. FOX：A FOX-Inspired Optimization Algorithm [J]. Applied Intelligence，2023，53(1)：1030-1050.

[7] HU G，GUO Y X，WEI G，et al. Genghis Khan Shark Optimizer：A Novel Nature-Inspired Algorithm for Engineering Optimization [J]. Advanced Engineering Informatics，2023，58：102210-1-102210-36.

[8] WU G，MALLIPEDDI R，SUGANTHAN P. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization [R]. Changsha：National University of Defense Technology，2017.

[9] ABDEL-BASSET M，MOHAMED R，ABOUHAWWASH M. Crested Porcupine Optimizer：A New Nature-Inspired Metaheuristic [J]. Knowledge-Based Systems，2024，284：111257-1-111257-42.

(责任编辑：韩　啸)