一种新的启发式优化算法 — 五行环优化算法研究与分析

刘漫丹 [1]

摘 要 借鉴中国古代哲学理论所描述的系统动态平衡方法, 提出了解决连续函数优化问题的五行环优化算法. 首先, 分析了基于五行元素生克原理而建立的五行环模型, 并在该模型基础上, 构建了元素空间结构以及元素更新方法等关键环节, 从而实现了五行环优化算法. 随后, 对五行环优化算法进行了性能分析和关键参数比较, 针对标准测试函数, 将五行环优化算法与其他 17 个机制各异的启发式优化算法进行了比较, 实验结果验证了五行环优化算法的有效性和通用性, 也表明了其在求解连续函数优化问题上具有较好的优化性能.

关键词 连续函数优化, 五行环优化算法, 启发式算法, 标准测试函数

引用格式 刘漫丹. 一种新的启发式优化算法 — 五行环优化算法研究与分析. 自动化学报, 2020, **46**(5): 957−970

DOI 10.16383/j.aas.c170657

Research and Analysis of a Novel Heuristic Algorithm: Five-elements Cycle Optimization Algorithm

LIU Man-Dan[1]

Abstract The five-elements cycle optimization algorithm (FECO) for continuous optimization problems is researched and analyzed in this paper. It is inspired by the theory of Five-elements which represents the performance of a dynamic balancing system. Firstly, the five-elements cycle model based on the mechanism of generation and restriction among five elements is analyzed. Afterwards, FECO is built for finding the optimal solution of continuous functions by designing the framework of element space and the pivotal operators. The performance and parameter comparison of FECO is given by experiment, the comparison with 17 optimization algorithms based on various mechanisms for two sets of benchmark functions is also given, which indicates the feasibility and universality of FECO.

Key words Continuous optimization, five-elements cycle optimization, heuristic algorithms, benchmark functions

Citation Liu Man-Dan. Research and analysis of a novel heuristic algorithm: five-elements cycle optimization algorithm. *Acta Automatica Sinica*, 2020, **46**(5): 957−970

优化问题是工程领域普遍存在且需要解决的问题. 最优化方法运用数学方法实现多种搜索策略, 使目标函数达到最优. 但是对于复杂工程问题, 由于受到计算时间、算法条件等因素的限制, 很难得到问题的最优解. 启发式优化方法则基于问题的直观或先验知识设计各种搜索策略, 获取问题的满意解, 对于解决复杂工程优化问题具有实际意义. 大多启发式优化方法源于自然系统、生物系统中一些现象的启示和提炼, 如模拟退火算法[1]、遗传算法[2]、蚁群优化算法[3]、粒子群优化算法[4], 以及各种新兴的群智能算法及其改进, 如人工蜂群算法[5]、萤火虫算法[6]、灰狼优化算法[7]、斑蝶优化算法[8]、鲸鱼优化算法[9]、花朵授粉算法[10]、斑鬣狗优化算法[11] 等, 这些算法已成功应用于各应用领域[12], 解决了各种不同的优化问题. 研究者们仍在不断寻求自然界中的隐喻机制, 以获取性能更佳的启发式优化方法.

中国古代哲学思想中的阴阳五行学说为人们提供了一种描述系统的方法, 这一学说试图以自然力量来解释自然现象, 代表了一种科学探索的倾向[13]. 阴阳五行学说将一个系统中的所有元素赋予阴、阳, 以及金、水、木、火、土的属性, 不同属性的元素之间具有不同的作用方式, 这些作用方式相互关联、相互竞争、相互协助, 从而使得系统具有动态平衡的特性, 系统的能量函数随时间变化, 最终达到最小. 这一现象启示我们将这种作用方式进行抽象、提炼, 形成一种算法用来求解优化问题. 已有一些研究者尝试着实现这个目标, Tam 等[14] 借鉴《易经》中六爻表示方法提出了 AOC (Algorithm of changes) 算法, 用于解决装箱问题; Zhao[15] 提出了 YYO (Yin-yang optimization) 算法, 该算法通过平衡个体的阴、阳属性, 使其定义的和谐函数趋于 0, 从而实现优化问题的求解; Cui 等[16] 提出了 NFESA (Naïve five-element string algorithm)

收稿日期 2017-11-20 录用日期 2018-07-05
Manuscript received November 20, 2017; accepted July 5, 2018
本文责任编委 乔俊飞
Recommended by Associate Editor QIAO Jun-Fei
1. 华东理工大学信息科学与工程学院 上海 200237
1. School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237

算法, 该算法借鉴五行相生原理, 采用五进制对个体进行编码, 并基于五进制基因型表达衍生出更多候选解, 从而提高搜索效率, 该算法用于求解连续函数优化问题; Punnathanam 等[17] 提出了 YYPO (Yin-yang-pair optimizer) 算法, 该算法基于种群空间中代表对立面阴和阳的两个解进行搜索, 这两个解分别表达了对搜索空间的空间探索性和局域挖掘, 通过分裂操作和存档操作, 实现空间探索与局域挖掘之间的平衡, 算法用于求解连续函数优化问题, 获得了满意的优化性能.

本文作者在文献 [18] 中提出了用于解决旅行商问题 (TSP) 的五行环优化算法 (Five-elements cycle optimization, FECO), 文献 [18] 中首次提出了五行环模型 (Five-elements cycle model, FECM), 五行环模型建立在五行相生相克原理的基础上, 通过计算个体之间的相互作用关系, 衡量出个体适应值的优劣, 随后根据个体的优劣进行解的更新, 通过迭代找出旅行商问题的最好解. 本文在文献 [18] 和 [19] 的五行环模型基础上增加了权重系数, 并进行了模型的数值仿真和收敛性验证, 基于该五行环模型, 本文提出了新的解决连续函数优化问题的五行环优化算法, 通过对比实验验证算法的有效性.

1 五行环模型 (FECM) 的建立与性能分析

中国古代哲学思想中的五行元素理论描述了动态系统的平衡关系. 五行元素指的是金、水、木、火、土, 它们之间存在着相生相克的关系, 如图 1 所示.

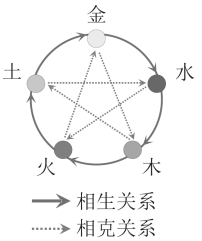


图 1 五行元素的相生相克关系

Fig. 1 The generating and restricting interactions among five elements

相生关系表示为: 金生水, 水生木, 木生火, 火生土, 土生金. 相生关系类比于母子关系, 子元素从母元素处获取养分从而使自己强大. 相克关系表示为: 木克土, 土克水, 水克火, 火克金, 金克木. 相克关系类比于祖孙关系, 孙元素被祖辈元素约束挟制, 从而使自己的生长受到制约. 通过相生相克关系, 所有的元素既相互制约又相互促进, 形成一种动态平衡关系.

假设一个动态系统由五个元素组成, 定义 k 时刻五个元素的质量分别为 $m_i(k)$ ($m_i(k) > 0$), 其中, $m_1(k)$ 代表金元素的质量, $m_2(k)$ 代表水元素的质量, $m_3(k)$ 代表木元素的质量, $m_4(k)$ 代表火元素的质量, $m_5(k)$ 代表土元素的质量. 在 k 时刻, 每个元素受到的力定义为 $F_i(k)$, $F_i(k)$ 是其他四个元素施加到元素 i 上的力.

首先, 以木元素 (即 $i = 3$) 为例, 说明 $m_i(k)$ 和 $F_i(k)$ 之间的关系. 可将 $F_3(k)$ 分解为四个部分: 第一部分记为 $F_{3\_1}(k)$, 是其母元素 (水) 向其施加的 "生" 力, 因来自母元素的 "生" 力将使该元素变强壮, 因此定义这个力为正值. $F_{3\_1}(k)$ 的大小与当前时刻的木元素质量 ($m_3(k)$) 和水元素质量 ($m_2(k)$) 相关, 当 $m_2(k) \geq m_3(k)$ 时, 木元素在 "生" 力作用下变得更强壮, 且 $m_2(k)$ 越大, 或者 $m_3(k)$ 越小, 则 $F_{3\_1}(k)$ 的值越大. 然而, 如果木元素本身比其母元素更强壮, 即 $m_2(k) < m_3(k)$, 那么其母元素将受到损伤, 因此 $F_{3\_1}(k)$ 的符号将会反转. 采用自然对数函数来表达以上的关系, 即 $F_{3\_1}(k)$ 可表示为式 (1).

$$F_{3\_1}(k) = \ln\left[\frac{m_2(k)}{m_3(k)}\right] \tag{1}$$

$F_3(k)$ 的第二部分记为 $F_{3\_2}(k)$, 这部分力来自其祖辈元素 (金) 的 "克" 力, 该力应为负值, 因为 "克" 力将使该元素变得衰弱. 同样地, $F_{3\_2}(k)$ 的值取决于木元素质量 ($m_3(k)$) 和金元素质量 ($m_1(k)$), 当 $m_1(k) \geq m_3(k)$ 时, 木元素在 "克" 力作用下变得更衰弱, 且 $m_1(k)$ 越大, 或者 $m_3(k)$ 越小, 则 $F_{3\_2}(k)$ 的绝对值越大. 如果木元素本身比其祖辈元素更强壮, 即 $m_1(k) < m_3(k)$, 那么 $F_{3\_2}(k)$ 的符号将反转. $F_{3\_2}(k)$ 可表示为式 (2).

$$F_{3\_2}(k) = -\ln\left[\frac{m_1(k)}{m_3(k)}\right] \tag{2}$$

同样地, $F_3(k)$ 的第三部分记为 $F_{3\_3}(k)$, 它是木元素施加给其子元素 (火) 的 "生" 力, 木元素在施加力的过程中损耗了自身, 因此该力应为负值. $F_3(k)$ 的第四部分记为 $F_{3\_4}(k)$, 它是木元素施加给其孙辈元素 (土) 的 "克" 力, 同样木元素在施加力的过程中也损耗了自身, 因此该力也应为负值. $F_{3\_3}(k)$ 和 $F_{3\_4}(k)$ 可表示为式 (3) 和式 (4).

$$F_{3\_3}(k) = -\ln\left[\frac{m_3(k)}{m_4(k)}\right] \tag{3}$$

$$F_{3\_4}(k) = -\ln\left[\frac{m_3(k)}{m_5(k)}\right] \tag{4}$$

将 $F_3(k)$ 表示为上述四个部分的叠加, 考虑到四个部分的作用并非完全相同, 定义 w_{gp}, w_{rp}, w_{ga} 和 w_{ra} 分别为上述四个部分的权重系数, 这些权重

应为 [0,1] 区间的正实数. 同时, 将 $F_3(k)$ 推演至 $F_i(k)$ $(i=1,2,3,4,5)$, 则得到式 (5).

$$\begin{cases} F_1(k)=w_{gp}\ln\left[\dfrac{m_5(k)}{m_1(k)}\right]-w_{rp}\ln\left[\dfrac{m_4(k)}{m_1(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_1(k)}{m_2(k)}\right]-w_{ra}\ln\left[\dfrac{m_1(k)}{m_3(k)}\right] \\ F_2(k)=w_{gp}\ln\left[\dfrac{m_1(k)}{m_2(k)}\right]-w_{rp}\ln\left[\dfrac{m_5(k)}{m_2(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_2(k)}{m_3(k)}\right]-w_{ra}\ln\left[\dfrac{m_2(k)}{m_4(k)}\right] \\ F_3(k)=w_{gp}\ln\left[\dfrac{m_2(k)}{m_3(k)}\right]-w_{rp}\ln\left[\dfrac{m_1(k)}{m_3(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_3(k)}{m_4(k)}\right]-w_{ra}\ln\left[\dfrac{m_3(k)}{m_5(k)}\right] \\ F_4(k)=w_{gp}\ln\left[\dfrac{m_3(k)}{m_4(k)}\right]-w_{rp}\ln\left[\dfrac{m_2(k)}{m_4(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_4(k)}{m_5(k)}\right]-w_{ra}\ln\left[\dfrac{m_4(k)}{m_1(k)}\right] \\ F_5(k)=w_{gp}\ln\left[\dfrac{m_4(k)}{m_5(k)}\right]-w_{rp}\ln\left[\dfrac{m_3(k)}{m_5(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_5(k)}{m_1(k)}\right]-w_{ra}\ln\left[\dfrac{m_5(k)}{m_2(k)}\right] \end{cases} \tag{5}$$

每个元素在力的作用下, 其质量由 $m_i(k)$ 变为 $m_i(k+1)$, 变化规则由式 (6) 表达.

$$m_i(k+1)=m_i(k)\cdot f_M(F_i(k)),\ \ i=1,2,3,4,5 \tag{6}$$

式 (6) 中, $f_M(\cdot)$ 为一非线性函数, 用来表达元素在受力作用下其质量的变化规律, 例如, 可采用 *logsig* 函数, 同时将式 (5) 和 (6) 中的五行元素关系拓展至 L 元素关系, 如式 (7) 所示, 式 (7) 即为五行环模型 (Five-elements cycle model, FECM).

$$\begin{cases} F_i(k)=w_{gp}\ln\left[\dfrac{m_{i-1}(k)}{m_i(k)}\right]-w_{rp}\ln\left[\dfrac{m_{i-2}(k)}{m_i(k)}\right]- \\ \qquad w_{ga}\ln\left[\dfrac{m_i(k)}{m_{i+1}(k)}\right]-w_{ra}\ln\left[\dfrac{m_i(k)}{m_{i+2}(k)}\right] \\ m_i(k+1)=m_i(k)\cdot\dfrac{2}{1+\exp(-F_i(k))} \end{cases} \tag{7}$$

上式中, $i=1,2,\cdots,L$. 当 $i=1$ 时, 用 "L" 替代 "$i-1$", 用 "$L-1$" 替代 "$i-2$"; 当 $i=2$ 时, 用 "L" 替代 "$i-2$"; 当 $i=L-1$ 时, 用 "1" 替代 "$i+2$"; 当 $i=L$ 时, 用 "1" 替代 "$i+1$", 用 "2" 替代 "$i+2$".

为了测试五行环模型的收敛性, 在 MATLAB 环境下对式 (7) 进行了实现. 对于式 (7) 的模型, 将各元素的质量初始值 $m_i(0)$ 和受力初始值 $F_i(0)$ 设置为 (0, 1] 区间的随机数. 首先, 将五行环模型中的权重系数设置为 $w_{gp}=w_{rp}=w_{ga}=w_{ra}=1$; 将元素数量 L 设置为 5. 各元素的质量 $m_i(k)$ 和受力 $F_i(k)$ 随时间变化的曲线如图 2 所示, 可见 $m_i(k)$ 和 $F_i(k)$ 在一定迭代次数后达到了稳定.

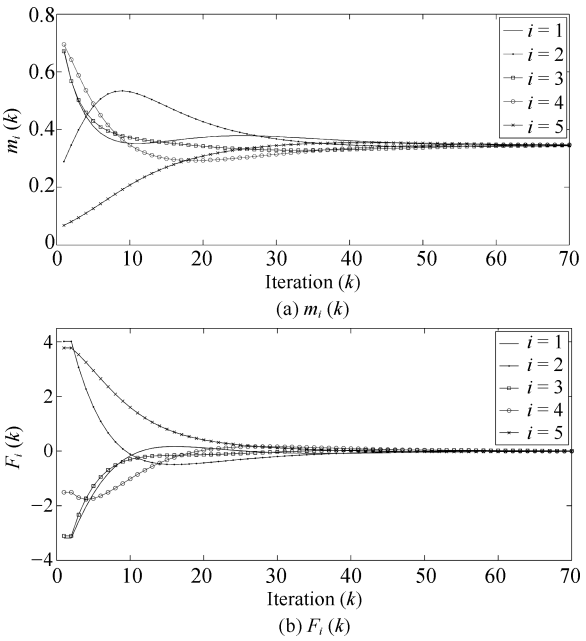
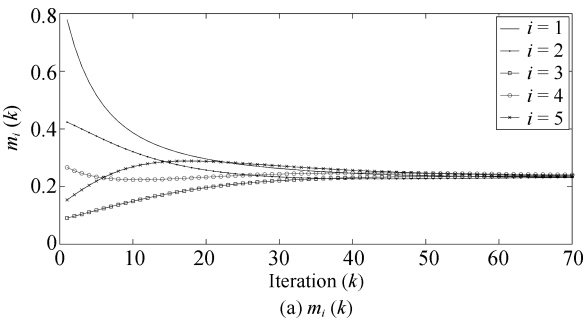


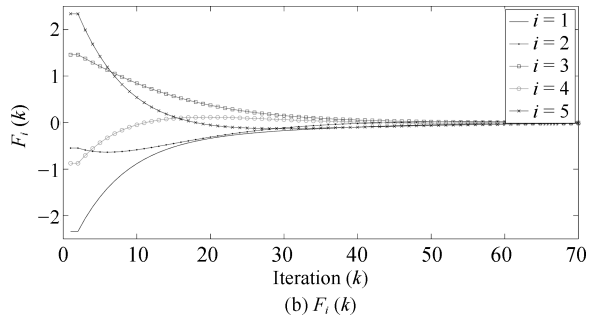
图 2　当 $w_{gp}=w_{rp}=w_{ga}=w_{ra}=1$, $L=5$ 时, $m_i(k)$ 和 $F_i(k)$ 的变化曲线

Fig. 2　The changing curves of $m_i(k)$, $F_i(k)$ when $w_{gp}=w_{rp}=w_{ga}=w_{ra}=1$, $L=5$

如果将权重系数 w_{gp}, w_{rp}, w_{ga} 和 w_{ra} 在每一迭代中都随机设置为 (0, 1] 区间的数值, 仍可得到一个收敛的五行环模型, 如图 3 所示.

实际上, 当 $L>5$ 时, 五行环模型仍是一个收敛模型. 图 4 显示了当 $L=20$, $w_{gp}=w_{rp}=w_{ga}=w_{ra}=1$ 时, 分别在第 1 代、第 50 代、第 300 代、第 1 000 代, 20 个元素的质量分布情况. 可以看出, 在迭代初期, 各元素质量各不相同, 而在迭代后期, 各元素的质量则逐渐趋同.

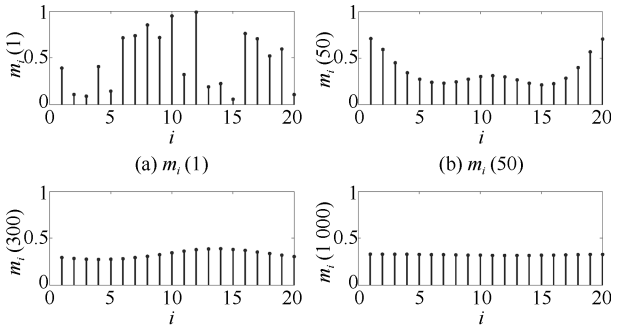




(b) $F_i(k)$

图 3 当 $w_{gp}, w_{rp}, w_{ga}, w_{ra}$ 为 (0, 1] 区间上的随机数, $L = 5$ 时, $m_i(k)$ 和 $F_i(k)$ 的变化曲线

Fig. 3 The changing curves of $m_i(k)$, $F_i(k)$ when w_{gp}, w_{rp}, w_{ga}, w_{ra} is random numerical value in (0, 1] and $L = 5$



(a) $m_i(1)$ (b) $m_i(50)$ (c) $m_i(300)$ (d) $m_i(1\,000)$

图 4 元素质量 $m_i(k)$ $(i = 1, 2, \cdots, 20)$ 分别在第 1、50、300、1 000 代时的变化情况

Fig. 4 The changing of $m_i(k)$ $(i = 1, 2, \cdots, 20)$ at 1st, 50th, 300th, and 1 000th iterations

从上述几个测试实验可以看出, 式 (7) 表示的五行环模型在元素之间的相生相克作用下, 各元素质量可趋于平衡, 各元素之间的差异达到最小.

2 五行环优化算法 (FECO) 的实现

一个无约束的连续函数优化问题可表达为式 (8):

$$\min\ f(x) \quad \text{s.t.} \quad x \in \left[x_d^{(\text{low})}, x_d^{(\text{high})}\right]^D \tag{8}$$

其中, $x = (x_1, x_2, \cdots, x_D)^{\mathrm{T}} \in \mathbf{R}^D$ 为 D 维决策向量, $[x_d^{(\text{low})}, x_d^{(\text{high})}]^D$ $(d = 1, 2, \cdots, D)$ 是搜索空间的可行区域, $f : \mathbf{R}^D \to \mathbf{R}$ 为目标函数的映射.

基于五行环模型, 本文构建了一种五行环优化算法 (Five-elements cycle optimization, FECO) 来获取 $f(x)$ 的最优解.

2.1 解的表达与种群结构

将 $f(x)$ 的每一个解对应于五行环模型中的每一个元素, 在 FECO 算法中, 将所有元素 (即当前所有可行解) 划分为 q 个环, 每个环包含 L 个元素, 即种群规模为 $L \times q$.

用 $x_{ij}(k)$ 表示第 k 代时第 j 个环中的第 i 个元素, $x_{ij}(k)$ 即为 $f(x)$ 的一个解. $m_{ij}(k)$ 是元素 $x_{ij}(k)$ 的质量, 对应为目标函数 $f(x_{ij}(k))$ 的值. $x_{ij}(k)$ 受到的来自第 j 个环中的其他元素的作用力记为 $F_{ij}(k)$, 用式 (9) 表达.

$$\begin{aligned} F_{ij}(k) = {} & w_{gp} \ln\left[\frac{m_{(i-1)j}(k)}{m_{ij}(k)}\right] - w_{rp} \ln\left[\frac{m_{(i-2)j}(k)}{m_{ij}(k)}\right] - \\ & w_{ga} \ln\left[\frac{m_{ij}(k)}{m_{(i+1)j}(k)}\right] - w_{ra} \ln\left[\frac{m_{ij}(k)}{m_{(i+2)j}(k)}\right] \end{aligned} \tag{9}$$

FECO 是一个迭代算法. 当 $k = 0$ 时, 随机产生 $L \times q$ 个初始元素 $x_{ij}(0)$ $(i = 1, 2, \cdots, L;\ j = 1, 2, \cdots, q)$, 并计算它们的质量 (即目标函数), 表示为 $m_{ij}(0)$. 同时, 设置初始的 $F_{ij}(0)$ 均为 0. 在第 k 代时, 首先计算元素 $x_{ij}(k)$ 的质量 $m_{ij}(k)$, 并通过式 (9) 计算其受力 $F_{ij}(k)$. $F_{ij}(k)$ 的值表征了元素 $x_{ij}(k)$ 的优劣, 根据 $F_{ij}(k)$ 决定元素从 $x_{ij}(k)$ 更新至 $x_{ij}(k+1)$ 的方式. 随着迭代过程的进行, 较好的元素被保留, 较差的元素被替换, 当达到最大迭代代数时, 可获取优化问题的最好解.

2.2 元素的更新

$x_{ij}(k+1)$ 的更新依赖于 $F_{ij}(k)$. 如果 $F_{ij}(k) > 0$, 说明 $x_{ij}(k)$ 是一个较好解, 因为从系统平衡的角度分析, $m_{ij}(k)$ 越小, $F_{ij}(k)$ 才会表现出越大, 大的受力才能使该元素变得强壮, 从而使各元素质量进一步趋同. 在这种情况下, $x_{ij}(k)$ 应该保留在解空间, 因此:

$$x_{ij}(k+1) = x_{ij}(k), \quad \text{若 } F_{ij}(k) > 0 \tag{10}$$

如果 $F_{ij}(k) \le 0$, 说明 $x_{ij}(k)$ 可能不是一个较好解, 因为其受力 $F_{ij}(k)$ 为负, 表明系统期望该元素变得更加衰弱, 也即说明其本身的质量 $m_{ij}(k)$ 比较大. 因 $x_{ij}(k)$ 是一个 D 维向量, 将其表示为 $[x_{ij,1}(k), x_{ij,2}(k), \cdots, x_{ij,d}(k), \cdots, x_{ij,D}(k)]^{\mathrm{T}}$. 在这种情况下, 应该对 $x_{ij}(k)$ 进行替换, 而且应该在较好解的邻域内产生新解, 为此将 $x_{ij,d}(k+1)$ 按照下式进行更新:

$$\begin{cases} x_{ij,d}(k+1) = x_{i^*j,d}(k) + r_s \times p_s \times [x_{i^*j,d}(k) - x_{ij,d}(k)], & \text{若 } r_m < p_m \\ x_{ij,d}(k+1) = x_{\text{best},d}(k) + r_s \times p_s \times [x_{\text{best},d}(k) - x_{i^*j,d}(k)], & \text{若 } r_m \ge p_m \end{cases} \tag{11}$$

上式中, p_s 是一个尺度因子, p_m 是一个预先给定的概率值; r_s 是 $[-1,1]$ 区间的随机数, r_m 是 [0,1] 区间的随机数; $x_{i^*j}(k)$ 是第 j 个环中受力最大的元素, 即 $F_{i^*j}(k) \geq F_{ij}(k)$ $(i^* \in \{1,2,\cdots,L\}, i\forall\{1,2,\cdots,L\})$, $x_{i^*j}(k) = [x_{i^*j,1}(k), x_{i^*j,2}(k), \cdots, x_{i^*j,d}(k), \cdots, x_{i^*j,D}(k)]^{\mathrm{T}}$; x_{best} 是当前最优解, $x_{\mathrm{best}} = [x_{\mathrm{best},1}, x_{\mathrm{best},2}, \cdots, x_{\mathrm{best},d}, \cdots, x_{\mathrm{best},D}]^{\mathrm{T}}$.

式 (11) 的含义是, $x_{ij}(k)$ 在给定的概率 p_m 下, 被替换为第 j 个环中受力定义下的最好元素 $x_{i^*j}(k)$ 的变异解, 在 $1-p_m$ 概率下, $x_{ij}(k)$ 被替换为当前全局最优解 x_{best} 的变异解.

2.3 FECO 算法的流程与步骤

FECO 算法的流程图如图 5 所示. 初始化阶段包括设置初始参数 L、q、最大迭代代数 $max\ iteration$、尺度因子 p_s、给定概率 p_m, 以及权重系数 w_{gp}, w_{rp}, w_{ga} 和 w_{ra}. 设置或计算初始值 $x_{ij}(0)$, $m_{ij}(0)$ 和 $F_{ij}(0)$. 通常情况下, L 的典型取值为 5, q 可取 $10\sim100$, p_s 可取 $0.1\sim2.0$, p_m 应在 [0, 1] 区间取值, 权重系数可简单地设为 $w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$. 对于参数的具体取值可参考下文的算法参数比较实验.

在第 k 代时, 对于第 j 个环中的元素, 它们的受力 $F_{ij}(k)$ $(i = 1,2,\cdots,L)$ 根据 $m_{ij}(k)$ 进行计算, 而 $m_{ij}(k)$ 则根据元素 $x_{ij}(k)$ 计算得到. 根据 $F_{ij}(k)$, 决定采用式 (10) 还是式 (11) 对元素进行更新, 得到 $x_{ij}(k+1)$. 在迭代过程中, 记录下当前最好解 x_{best}, 直至 $k \geq max\ iteration$ 时.

3 五行环优化算法 (FECO) 的性能分析与参数比较

根据上述 FECO 算法的原理和实现步骤, 在 MATLAB R2014a 环境下对 FECO 算法进行编程实现, 并对 FECO 算法的性能进行分析和参数比较.

3.1 测试函数

为了分析算法性能并比较算法参数, 采用文献 [20] 中的测试函数集进行实验, 共有 23 个函数 $(f_1 - f_{23})$, 其中, $f_1 - f_{13}$ 为高维函数 $(D = 30)$, $f_{14} - f_{23}$ 为低维函数, 维度分别为 2、3、4 或 6. 根据文献 [20] 的分类, 将这些测试函数分为三类: 1) $f_1 - f_7$ 为单峰函数, 只有一个极值; 2) $f_8 - f_{13}$ 为多峰函数, 且局部极值数量随着函数维数的增加呈指数趋势增加; 3) $f_{14} - f_{23}$ 为低维、多峰函数, 具有若干局部极值.

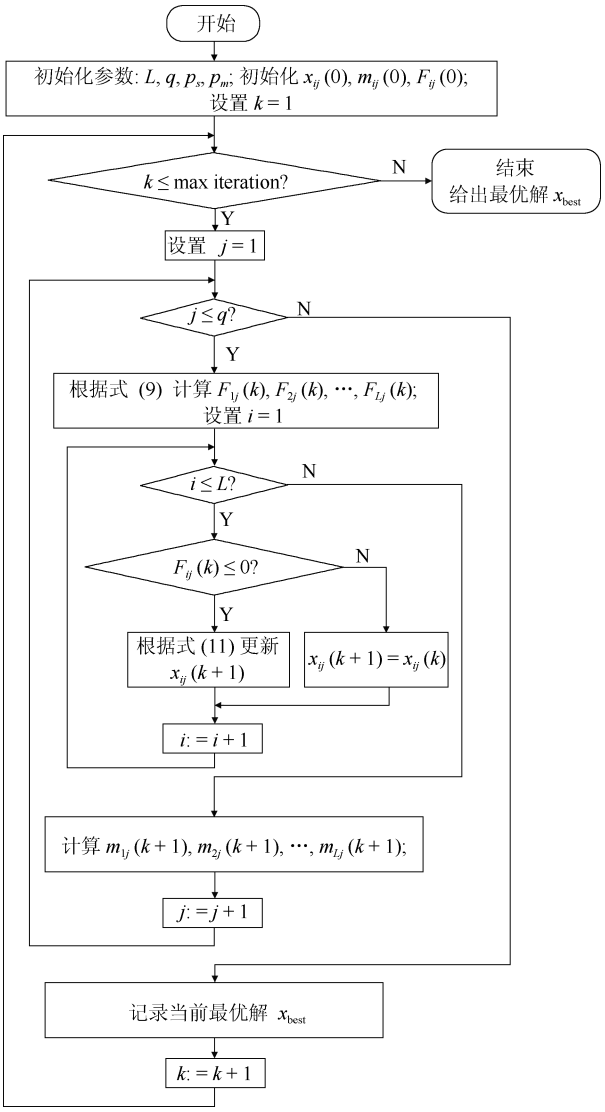


图 5　FECO 算法流程图

Fig. 5　The flowchart of FECO algorithm

3.2 FECO 算法的性能分析

为了能够更直观地显示 FECO 算法的工作过程, 以一个 2 维测试函数 Six-hump camel-back function (f_{16}) 为例对 FECO 算法进行分析. 该函数的决策变量 x 的定义域为 $[-5,5] \times [-5,5]$, 函数的最优解有 2 个, 分别为 (0.0898, −0.7126) 和 (−0.0898, 0.7126), 函数最小值为 −1.0316.

FECO 算法的参数设置为: $L = 5$, $q = 20$, $p_s = 1.0$, $p_m = 0.9$, $w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$, $max\ iteration = 50$. FECO 算法在运行到第 15 代时, 找到了该函数的最小值 −1.0316. 图 6 显示了在不同迭代时间, 元素在解空间的分布情况, 以及当前最优解随迭代次数变化的收敛曲线.

在图 6 (a) ∼ 6 (e) 中, 菱形代表 $F_{ij}(k) > 0$ 的元素, 它们应是较好的解; 圆点代表 $F_{ij}(k) \leq 0$ 的元

素, 它们应是相对较差的解. 在 FECO 算法中, 评价解的好坏是相对的, 是通过同一个环中各元素相互作用力来评价的, 而各个环之间则是相互独立的. 因此, 在迭代初期, 如图 6 (a) 所示, 菱形和圆点相互参杂地分布; 随着进化过程的进行, 元素渐渐向最优区域集中, 从图 6 (c) 和 6 (d) 可以看出, 菱形相对圆点更接近该函数的实际最优解, 表明用 $F_{ij}(k)$ 来评价元素的好坏是符合实际情况的. 从图 6 (e) 可以看出, 在该函数的两个最优解 $(0.0898, -0.7126)$ 和 $(-0.0898, 0.7126)$ 位置处, 均有元素分布, 说明在保持解的多样性方面, FECO 算法具有一定的能力.

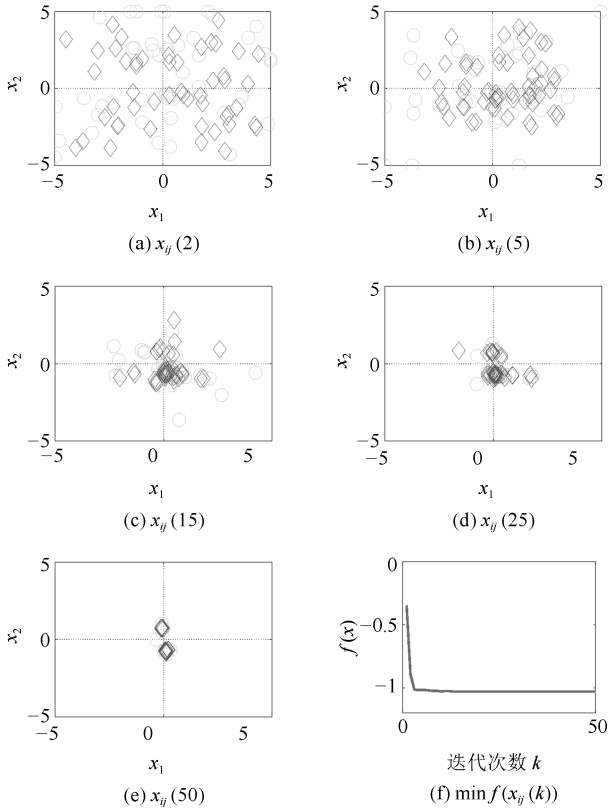


图 6 在不同迭代次数时的元素 $x_{ij}(k)$ 的分布情况以及最小 $f(x_{ij}(k))$ 的变化曲线

Fig. 6 The distribution of $x_{ij}(k)$ at various iterations and the changing curve of minimum $f(x_{ij}(k))$

3.3 FECO 算法的参数比较

在 FECO 算法中, 需要设置的参数主要有: L、q、p_s、p_m, 以及权重系数 w_{gp}, w_{rp}, w_{ga} 和 w_{ra}. 本文通过对 23 个测试函数的实验比较, 来选取合适的参数取值. 在参数比较实验中, 选取的一组基本参数值为: $L = 5$, $q = 20$, $p_s = 1.0$, $p_m = 0.9$, $w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$, 当对一个 (组) 参数进行比较时, 其他参数均选取上述的基本值. 表 1 ∼ 表 4 中显示的所有实验结果均是算法运行 51 次得到的最优函数值的平均结果, 其中, 最好结果加粗显示. 对于每个测试函数, 算法运行的目标函数计算次数 (function evaluations, FEs) 与文献 [20] 中给出的相同.

3.3.1 参数 L 和 q 的比较

$L \times q$ 的值相当于进化算法中的种群规模, 直观上分析, 较大的 q 值 (即将元素划分为较多的环) 有益于保持种群的多样性, 较大的 L 值 (即每个环中包含较多的元素) 有益于解的快速收敛, 但容易陷入局部极值.

表 1 给出了在 $L \times q = 100$ 情况下, 5 组 $L \sim q$ 取值的比较结果. 参照文献 [21] 中的对于进化算法结果评价的 Friedman 检验方法, 对 5 组参数取值进行比较, 实验得到的 p 值 (检验假设零假设成立的概率, 显著性水平取 0.05) 为 5.7740×10^{-14}, 表明这 5 组实验之间具有显著性差异. 其中, 最差的情况是 $L = 100$ 且 $q = 1$ 时, 表明将所有元素组成一个环的做法效果不好, 应该将元素划分为若干环, 而且小环 (即小 L 值) 优于大环 (即大 L 值). 根据表 1 的实验结果, 选取 $L = 5$, $q = 20$.

3.3.2 参数 p_s 的比较

参数 p_s 的取值决定了元素更新时的变异扩展程度, 从直观上分析, 较大的 p_s 值可以更快地找到更好解, 较小的 p_s 值有益于种群的稳定性, 但将导致收敛速度变慢.

表 2 给出了 9 组 p_s 取值的比较结果, 实验得到的 p 值为 2.1032×10^{-18}, 表明这 9 组实验之间具有显著性差异. 从表 2 可以看出, p_s 的取值偏大或偏小都影响优化效果, 对于所有的 23 个测试函数来说, p_s 取 1.0 较为合适.

3.3.3 参数 p_m 的比较

参数 p_m 的取值决定了元素更新时的探索区域, 直观上分析, 较大的 p_m 值有益于保持种群的多样性, 较小的 p_m 值可以快速收敛至当前最优解, 但容易陷入局部极值.

表 3 给出了 11 组 p_m 取值的比较结果, 实验得到的 p 值为 5.1039×10^{-3}, 表明这 11 组实验之间具有显著性差异. 其中, 最差的情况为 p_m 取 1.0 时, 表明在对元素进行更新时, 必须同时考虑局部最优区域和全局最优区域. 根据表 3 的实验结果, 选取 $p_m = 0.9$.

3.3.4 权重系数的比较

对于权重系数 w_{gp}, w_{rp}, w_{ga} 和 w_{ra}, 可将其全部设为 1, 也可以在每一次迭代中都设为 (0, 1] 区间的随机数. 表 4 给出了这两种情况的比较结果, 实验得到的 p 值为 3.9299×10^{-4}, 表明这两组实验之间具有显著性差异, 当 $w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$

表 1　用 Friedman 检验法 (显著性水平 0.05) 比较参数 L 和 q

Table 1　Comparison of L and q with Friedman test (significance value 0.05)

Function	$L=5/q=20$	$L=10/q=10$	$L=20/q=5$	$L=50/q=2$	$L=100/q=1$
f_1	3.22×10^{-23}	7.65×10^{-25}	$\mathbf{1.64}\times10^{-27}$	3.65×10^{-18}	1.99×10^{3}
f_2	3.18×10^{-16}	9.35×10^{-18}	$\mathbf{3.35}\times10^{-20}$	9.80×10^{-1}	2.88×10^{1}
f_3	$\mathbf{1.47}\times10^{2}$	6.88×10^{2}	1.35×10^{3}	2.62×10^{3}	1.22×10^{4}
f_4	$\mathbf{4.22}\times10^{-1}$	1.47×10^{0}	3.07×10^{0}	1.13×10^{1}	4.67×10^{1}
f_5	$\mathbf{5.29}\times10^{1}$	5.59×10^{1}	1.83×10^{3}	1.39×10^{2}	1.15×10^{6}
f_6	$\mathbf{0.00}\times10^{0}$	$\mathbf{0.00}\times10^{0}$	1.37×10^{-1}	1.11×10^{2}	4.61×10^{3}
f_7	1.27×10^{-2}	8.45×10^{-3}	$\mathbf{7.29}\times10^{-3}$	2.03×10^{-2}	2.56×10^{0}
f_8	$\mathbf{-1.15}\times10^{4}$	-1.14×10^{4}	-1.10×10^{4}	-9.85×10^{3}	-7.97×10^{3}
f_9	$\mathbf{1.23}\times10^{1}$	1.31×10^{1}	1.94×10^{1}	5.79×10^{1}	1.48×10^{2}
f_{10}	1.53×10^{-12}	$\mathbf{2.22}\times10^{-13}$	1.70×10^{-1}	3.60×10^{0}	1.42×10^{1}
f_{11}	$\mathbf{6.10}\times10^{-4}$	1.24×10^{-3}	4.34×10^{-3}	1.17×10^{-1}	1.80×10^{1}
f_{12}	$\mathbf{4.07}\times10^{-3}$	2.64×10^{-2}	7.53×10^{-2}	4.58×10^{-1}	5.21×10^{5}
f_{13}	$\mathbf{5.95}\times10^{-2}$	6.66×10^{-1}	2.21×10^{0}	1.32×10^{1}	1.99×10^{6}
f_{14}	$\mathbf{1.02}\times10^{0}$	1.06×10^{0}	1.14×10^{0}	1.37×10^{0}	2.50×10^{0}
f_{15}	$\mathbf{5.65}\times10^{-4}$	5.94×10^{-4}	6.19×10^{-4}	7.18×10^{-4}	1.22×10^{-3}
f_{16}	-1.03×10^{0}	-1.03×10^{0}	$\mathbf{-1.03}\times10^{0}$	-1.03×10^{0}	-1.03×10^{0}
f_{17}	3.98×10^{-1}	3.98×10^{-1}	$\mathbf{3.98}\times10^{-1}$	3.98×10^{-1}	3.98×10^{-1}
f_{18}	3.00×10^{0}	$\mathbf{3.00}\times10^{0}$	$\mathbf{3.00}\times10^{0}$	3.00×10^{0}	3.00×10^{0}
f_{19}	-3.86×10^{0}	$\mathbf{-3.86}\times10^{0}$	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}
f_{20}	-3.30×10^{0}	-3.31×10^{0}	$\mathbf{-3.32}\times10^{0}$	-3.29×10^{0}	-3.28×10^{0}
f_{21}	$\mathbf{-1.00}\times10^{1}$	-9.64×10^{0}	-8.37×10^{0}	-6.61×10^{0}	-4.69×10^{0}
f_{22}	$\mathbf{-9.99}\times10^{0}$	-9.84×10^{0}	-9.70×10^{0}	-7.07×10^{0}	-6.21×10^{0}
f_{23}	$\mathbf{-1.03}\times10^{1}$	-1.03×10^{1}	-8.62×10^{0}	-6.92×10^{0}	-4.93×10^{0}
Friedman 排序	**1.89**	1.91	2.41	3.78	5

表 2　用 Friedman 检验法 (显著性水平 0.05) 比较参数 p_s

Table 2　Comparison of p_s with Friedman test (significance value 0.05)

Function	$p_s=0.2$	$p_s=0.4$	$p_s=0.6$	$p_s=0.8$	$p_s=1.0$	$p_s=1.2$	$p_s=1.4$	$p_s=1.6$	$p_s=1.8$
f_1	3.73×10^{2}	1.04×10^{-28}	$\mathbf{2.76}\times10^{-40}$	4.57×10^{-33}	3.22×10^{-23}	1.11×10^{-13}	5.40×10^{-6}	7.63×10^{-1}	5.05×10^{2}
f_2	8.94×10^{-1}	$\mathbf{3.43}\times10^{-27}$	5.25×10^{-27}	3.75×10^{-22}	3.18×10^{-16}	2.06×10^{-10}	1.67×10^{-5}	6.46×10^{-1}	2.81×10^{1}
f_3	6.14×10^{3}	3.43×10^{3}	1.52×10^{3}	2.48×10^{2}	$\mathbf{1.47}\times10^{2}$	3.02×10^{3}	1.41×10^{4}	3.29×10^{4}	5.52×10^{4}
f_4	4.28×10^{1}	2.60×10^{1}	1.33×10^{1}	1.52×10^{0}	$\mathbf{4.22}\times10^{-1}$	2.12×10^{0}	1.03×10^{1}	3.12×10^{1}	4.95×10^{1}
f_5	1.46×10^{5}	1.80×10^{2}	8.00×10^{1}	$\mathbf{4.79}\times10^{1}$	5.29×10^{1}	7.48×10^{1}	1.92×10^{3}	8.76×10^{3}	4.36×10^{5}
f_6	3.85×10^{2}	4.76×10^{0}	1.96×10^{-1}	3.92×10^{-2}	$\mathbf{0.00}\times10^{0}$	$\mathbf{0.00}\times10^{0}$	$\mathbf{0.00}\times10^{0}$	3.27×10^{0}	6.33×10^{2}
f_7	3.49×10^{-1}	7.34×10^{-2}	3.53×10^{-2}	2.24×10^{-2}	1.27×10^{-2}	$\mathbf{9.89}\times10^{-3}$	3.69×10^{-2}	1.38×10^{-1}	1.71×10^{0}
f_8	-1.17×10^{4}	$\mathbf{-1.18}\times10^{4}$	-1.17×10^{4}	-1.15×10^{4}	-1.15×10^{4}	-1.10×10^{4}	-1.04×10^{4}	-9.75×10^{3}	-8.78×10^{3}
f_9	2.03×10^{1}	1.04×10^{1}	$\mathbf{1.04}\times10^{1}$	1.08×10^{1}	1.23×10^{1}	1.37×10^{1}	1.83×10^{1}	4.22×10^{1}	1.48×10^{2}
f_{10}	4.97×10^{0}	1.22×10^{0}	5.48×10^{-2}	$\mathbf{1.39}\times10^{-14}$	1.53×10^{-12}	1.31×10^{-7}	3.92×10^{-1}	6.94×10^{0}	1.83×10^{1}
f_{11}	3.83×10^{0}	1.72×10^{-2}	1.78×10^{-3}	5.73×10^{-4}	6.10×10^{-4}	$\mathbf{6.34}\times10^{-6}$	3.03×10^{-3}	8.30×10^{-1}	5.52×10^{0}
f_{12}	1.21×10^{3}	3.14×10^{-1}	5.90×10^{-2}	8.13×10^{-3}	$\mathbf{4.07}\times10^{-3}$	1.63×10^{-2}	4.30×10^{-3}	4.53×10^{0}	8.29×10^{4}
f_{13}	1.15×10^{5}	1.55×10^{1}	2.98×10^{0}	7.40×10^{-2}	5.95×10^{-2}	$\mathbf{4.78}\times10^{-3}$	3.13×10^{0}	5.90×10^{1}	5.70×10^{5}
f_{14}	1.91×10^{0}	1.19×10^{0}	1.04×10^{0}	1.10×10^{0}	1.02×10^{0}	$\mathbf{1.02}\times10^{0}$	1.08×10^{0}	1.04×10^{0}	1.04×10^{0}
f_{15}	9.93×10^{-4}	6.09×10^{-4}	5.93×10^{-4}	5.50×10^{-4}	5.65×10^{-4}	$\mathbf{5.46}\times10^{-4}$	6.20×10^{-4}	6.55×10^{-4}	7.44×10^{-4}
f_{16}	$\mathbf{-1.03}\times10^{0}$	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.02×10^{0}	-1.02×10^{0}	-1.01×10^{0}
f_{17}	3.98×10^{-1}	3.98×10^{-1}	$\mathbf{3.98}\times10^{-1}$	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}
f_{18}	3.00×10^{0}	$\mathbf{3.00}\times10^{0}$	$\mathbf{3.00}\times10^{0}$	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}
f_{19}	$\mathbf{-3.86}\times10^{0}$	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}
f_{20}	-3.27×10^{0}	-3.28×10^{0}	-3.27×10^{0}	-3.29×10^{0}	$\mathbf{-3.30}\times10^{0}$	-3.29×10^{0}	-3.27×10^{0}	-3.24×10^{0}	-3.23×10^{0}
f_{21}	-8.15×10^{0}	-9.38×10^{0}	-9.66×10^{0}	-9.99×10^{0}	$\mathbf{-1.00}\times10^{1}$	-9.90×10^{0}	-9.93×10^{0}	-9.94×10^{0}	-9.24×10^{0}
f_{22}	-7.29×10^{0}	-9.42×10^{0}	-9.75×10^{0}	-1.02×10^{1}	-9.99×10^{0}	$\mathbf{-1.04}\times10^{1}$	-1.01×10^{1}	-9.99×10^{0}	-9.93×10^{0}
f_{23}	-8.11×10^{0}	-9.00×10^{0}	-1.02×10^{1}	-9.89×10^{0}	-1.03×10^{1}	-1.03×10^{1}	-1.02×10^{1}	$\mathbf{-1.04}\times10^{1}$	-9.56×10^{0}
Friedman 排序	7.04	4.78	3.65	3.04	**2.65**	3.39	5.30	6.61	8.52

表 3 用 Friedman 检验法 (显著性水平 0.05) 比较参数 p_m

Table 3 Comparison of p_m with Friedman test (significance value 0.05)

Function	$p_m = 0.0$	$p_m = 0.1$	$p_m = 0.2$	$p_m = 0.3$	$p_m = 0.4$	$p_m = 0.5$
f_1	9.97×10^{3}	3.43×10^{3}	5.14×10^{2}	5.55×10^{0}	7.19×10^{-26}	$\mathbf{2.80} \times 10^{-54}$
f_2	6.45×10^{1}	4.08×10^{1}	1.82×10^{1}	4.30×10^{0}	9.80×10^{-1}	3.92×10^{-1}
f_3	2.91×10^{4}	2.22×10^{4}	1.58×10^{4}	7.56×10^{3}	2.82×10^{3}	2.42×10^{3}
f_4	5.37×10^{1}	4.72×10^{1}	4.00×10^{1}	2.87×10^{1}	7.44×10^{0}	8.10×10^{-1}
f_5	8.71×10^{6}	2.01×10^{6}	2.25×10^{5}	6.37×10^{3}	2.00×10^{3}	1.87×10^{3}
f_6	1.20×10^{4}	5.37×10^{3}	1.58×10^{3}	3.35×10^{2}	7.03×10^{1}	1.51×10^{1}
f_7	7.62×10^{0}	2.64×10^{0}	1.28×10^{0}	5.08×10^{-1}	2.24×10^{-1}	1.45×10^{-1}
f_8	-7.10×10^{3}	-7.71×10^{3}	-8.25×10^{3}	-8.69×10^{3}	-8.81×10^{3}	-9.31×10^{3}
f_9	1.89×10^{2}	1.58×10^{2}	1.39×10^{2}	1.08×10^{2}	9.11×10^{1}	7.68×10^{1}
f_{10}	1.70×10^{1}	1.44×10^{1}	1.10×10^{1}	6.69×10^{0}	4.13×10^{0}	2.45×10^{0}
f_{11}	9.11×10^{1}	3.01×10^{1}	6.56×10^{0}	9.69×10^{-1}	5.39×10^{-2}	1.93×10^{-2}
f_{12}	5.70×10^{6}	8.31×10^{5}	2.34×10^{3}	4.86×10^{0}	7.95×10^{-1}	5.19×10^{-1}
f_{13}	2.39×10^{7}	3.24×10^{6}	8.92×10^{4}	6.24×10^{1}	6.66×10^{0}	2.05×10^{0}
f_{14}	1.94×10^{0}	2.02×10^{0}	1.41×10^{0}	1.11×10^{0}	1.09×10^{0}	$\mathbf{9.98} \times 10^{-1}$
f_{15}	6.63×10^{-3}	3.32×10^{-3}	3.82×10^{-3}	2.99×10^{-3}	3.22×10^{-3}	2.02×10^{-3}
f_{16}	-1.03×10^{0}	$\mathbf{-1.03} \times 10^{0}$	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}
f_{17}	$\mathbf{3.98} \times 10^{-1}$	$\mathbf{3.98} \times 10^{-1}$	$\mathbf{3.98} \times 10^{-1}$	3.98×10^{-1}	$\mathbf{3.98} \times 10^{-1}$	3.98×10^{-1}
f_{18}	3.53×10^{0}	3.00×10^{0}	3.53×10^{0}	3.00×10^{0}	$\mathbf{3.00} \times 10^{0}$	$\mathbf{3.00} \times 10^{0}$
f_{19}	$\mathbf{-3.86} \times 10^{0}$	$\mathbf{-3.86} \times 10^{0}$	$\mathbf{-3.86} \times 10^{0}$	$\mathbf{-3.86} \times 10^{0}$	$\mathbf{-3.86} \times 10^{0}$	-3.86×10^{0}
f_{20}	-3.26×10^{0}	-3.27×10^{0}	-3.28×10^{0}	-3.27×10^{0}	-3.27×10^{0}	-3.26×10^{0}
f_{21}	-5.73×10^{0}	-5.38×10^{0}	-5.43×10^{0}	-6.01×10^{0}	-5.10×10^{0}	-5.66×10^{0}
f_{22}	-6.02×10^{0}	-5.84×10^{0}	-6.17×10^{0}	-5.37×10^{0}	-6.82×10^{0}	-6.55×10^{0}
f_{23}	-5.27×10^{0}	-5.23×10^{0}	-5.98×10^{0}	-5.17×10^{0}	-6.45×10^{0}	-7.78×10^{0}
Friedman 排序	6.60	6.05	5.70	6.25	5.15	5.00

Function	$p_m = 0.6$	$p_m = 0.7$	$p_m = 0.8$	$p_m = 0.9$	$p_m = 1.0$
f_1	8.01×10^{-49}	1.87×10^{-40}	6.54×10^{-32}	3.22×10^{-23}	4.53×10^{4}
f_2	1.96×10^{-1}	$\mathbf{3.41} \times 10^{-27}$	7.72×10^{-22}	3.18×10^{-16}	1.03×10^{2}
f_3	2.12×10^{3}	1.53×10^{3}	1.08×10^{3}	$\mathbf{1.47} \times 10^{2}$	5.68×10^{4}
f_4	2.19×10^{-1}	$\mathbf{1.69} \times 10^{-1}$	2.25×10^{-1}	4.22×10^{-1}	7.67×10^{1}
f_5	9.59×10^{1}	$\mathbf{3.69} \times 10^{1}$	5.80×10^{1}	5.29×10^{1}	1.20×10^{8}
f_6	1.76×10^{0}	1.37×10^{0}	1.57×10^{-1}	$\mathbf{0.00} \times 10^{0}$	4.40×10^{4}
f_7	6.80×10^{-2}	4.88×10^{-2}	2.65×10^{-2}	$\mathbf{1.27} \times 10^{-2}$	5.67×10^{1}
f_8	-9.73×10^{3}	-1.00×10^{4}	-1.05×10^{4}	$\mathbf{-1.15} \times 10^{4}$	-4.61×10^{3}
f_9	5.27×10^{1}	3.90×10^{1}	2.66×10^{1}	$\mathbf{1.23} \times 10^{1}$	3.14×10^{2}
f_{10}	8.55×10^{-1}	2.13×10^{-1}	2.26×10^{-2}	$\mathbf{1.53} \times 10^{-12}$	1.97×10^{1}
f_{11}	9.21×10^{-3}	3.09×10^{-3}	9.18×10^{-4}	$\mathbf{6.10} \times 10^{-4}$	3.83×10^{2}
f_{12}	2.66×10^{-1}	2.20×10^{-1}	7.14×10^{-2}	$\mathbf{4.07} \times 10^{-3}$	2.24×10^{8}
f_{13}	2.45×10^{0}	9.56×10^{-1}	5.66×10^{-1}	$\mathbf{5.95} \times 10^{-2}$	5.39×10^{8}
f_{14}	1.04×10^{0}	9.98×10^{-1}	1.02×10^{0}	1.02×10^{0}	2.26×10^{0}
f_{15}	9.00×10^{-4}	1.13×10^{-3}	$\mathbf{4.26} \times 10^{-4}$	5.65×10^{-4}	4.09×10^{-3}
f_{16}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.02×10^{0}
f_{17}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}
f_{18}	3.00×10^{0}	$\mathbf{3.00} \times 10^{0}$	3.00×10^{0}	3.00×10^{0}	3.01×10^{0}
f_{19}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}
f_{20}	-3.25×10^{0}	-3.26×10^{0}	-3.26×10^{0}	$\mathbf{-3.30} \times 10^{0}$	-3.08×10^{0}
f_{21}	-6.61×10^{0}	-8.13×10^{0}	-9.10×10^{0}	$\mathbf{-1.00} \times 10^{1}$	-5.09×10^{0}
f_{22}	-7.66×10^{0}	-8.83×10^{0}	$\mathbf{-1.02} \times 10^{1}$	-9.99×10^{0}	-4.70×10^{0}
f_{23}	-7.84×10^{0}	-9.05×10^{0}	-9.78×10^{0}	$\mathbf{-1.03} \times 10^{1}$	-5.04×10^{0}
Friedman 排序	5.45	5.00	5.20	**4.90**	10.70

表 4　用 Friedman 检验法 (显著性水平 0.05) 比较权重系数 w_{gp}, w_{rp}, w_{ga}, w_{ra}

Table 4　Comparison of w_{gp}, w_{rp}, w_{ga}, w_{ra} with Friedman test (significance value 0.05)

Function	$w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$	w_{gp}, w_{rp}, w_{ga}, w_{ra} 为随机数
f_1	$\mathbf{3.22} \times 10^{-23}$	1.79×10^{-19}
f_2	$\mathbf{3.18} \times 10^{-16}$	5.71×10^{-14}
f_3	$\mathbf{1.47} \times 10^{2}$	7.04×10^{2}
f_4	$\mathbf{4.22} \times 10^{-1}$	8.05×10^{-1}
f_5	$\mathbf{5.29} \times 10^{1}$	5.91×10^{1}
f_6	$\mathbf{0.00} \times 10^{0}$	1.96×10^{-2}
f_7	$\mathbf{1.27} \times 10^{-2}$	1.52×10^{-2}
f_8	$\mathbf{-1.15} \times 10^{4}$	-1.10×10^{4}
f_9	$\mathbf{1.23} \times 10^{1}$	1.87×10^{1}
f_{10}	$\mathbf{1.53} \times 10^{-12}$	1.10×10^{-10}
f_{11}	6.10×10^{-4}	$\mathbf{2.17} \times 10^{-4}$
f_{12}	$\mathbf{4.07} \times 10^{-3}$	1.63×10^{-2}
f_{13}	$\mathbf{5.95} \times 10^{-2}$	1.88×10^{-1}
f_{14}	$\mathbf{1.02} \times 10^{0}$	1.08×10^{0}
f_{15}	5.65×10^{-4}	$\mathbf{5.45} \times 10^{-4}$
f_{16}	$\mathbf{-1.03} \times 10^{0}$	-1.03×10^{0}
f_{17}	$\mathbf{3.98} \times 10^{-1}$	3.98×10^{-1}
f_{18}	$\mathbf{3.00} \times 10^{0}$	3.00×10^{0}
f_{19}	$\mathbf{-3.86} \times 10^{0}$	-3.86×10^{0}
f_{20}	$\mathbf{-3.30} \times 10^{0}$	-3.27×10^{0}
f_{21}	$\mathbf{-1.00} \times 10^{1}$	-9.96×10^{0}
f_{22}	-9.99×10^{0}	$\mathbf{-1.00} \times 10^{1}$
f_{23}	$\mathbf{-1.03} \times 10^{1}$	-9.52×10^{0}
Friedman 排序	**1.13**	1.87

时, 优化性能更好. 对于一般的 FECO 算法, 可直接取 $w_{gp} = w_{rp} = w_{ga} = w_{ra} = 1$.

通过上述 FECO 算法参数分析, 将本文的参数设置如下: $L = 5$, $q = 20$, $p_s = 1.0$, $p_m = 0.9$. 表 5 给出了 FECO 算法优化 23 个测试函数时, 51 次独立运行中获得的最好结果、最差结果、平均结果和标准差.

4　五行环优化算法 (FECO) 与其他基于不同机制的启发式优化算法的比较

4.1　对 23 个测试函数的优化结果比较

针对上节中的 23 个测试函数, 将 FECO 算法与其他 15 种优化算法进行比较, 这些算法是基于不同的机制提出的, 包括遗传算法 (GA)、经典进化规划 (Classical evolutionary programming, CEP)[20]、快速进化规划 (Fast evolutionary programming, FEP)[20]、传统进化策略 (Conventional evolutionary strategy, CES)[22]、快速进化策略 (Fast evolutionary strategy, FES)[22]、差分进化算法 (DE)[23]、G3PCX 算法 (Generalized generation gap model with generic parent-centric recombination operator)[24]、粒子群优化算法 (PSO)、群搜索优化算法 (GSO)[25]、引力搜索算法 (Gravitational search algorithm, GSA)[26]、NFESA 算法 (Naïve five-element string algorithm)、实数编码的生物地理优化算法 (Real-coded biogeography-based optimization, RCBBO)[27]、实数编码的化学反应优化算法 (Real-coded chemical reaction optimization, RCCRO)[28]、灰狼优化算法 (Grey wolf optimizer, GWO), 以及鲸鱼优化算法 (Whale optimization algorithm, WOA).

表 5　FECO 优化 23 个测试函数的优化结果

Table 5　Optimization results of FECO for 23 benchmark functions

Function	最好 Best	最差 Worst	平均 Mean	标准差 StdDev
f_1	1.27×10^{-24}	1.15×10^{-22}	3.22×10^{-23}	2.71×10^{-23}
f_2	1.16×10^{-16}	9.84×10^{-16}	3.18×10^{-16}	1.68×10^{-16}
f_3	1.43×10^{1}	4.65×10^{2}	1.47×10^{2}	1.16×10^{2}
f_4	1.51×10^{-1}	1.59×10^{0}	4.22×10^{-1}	2.59×10^{-1}
f_5	1.18×10^{0}	1.91×10^{2}	5.29×10^{1}	3.84×10^{1}
f_6	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
f_7	4.17×10^{-3}	2.87×10^{-2}	1.27×10^{-2}	5.10×10^{-3}
f_8	-1.22×10^{4}	-1.09×10^{4}	-1.15×10^{4}	3.29×10^{2}
f_9	4.97×10^{0}	2.49×10^{1}	1.23×10^{1}	5.24×10^{0}
f_{10}	4.88×10^{-13}	4.58×10^{-12}	1.53×10^{-12}	8.44×10^{-13}
f_{11}	0.00×10^{0}	1.64×10^{-2}	6.10×10^{-4}	2.85×10^{-3}
f_{12}	7.25×10^{-24}	1.04×10^{-1}	4.07×10^{-3}	2.03×10^{-2}
f_{13}	8.50×10^{-20}	3.02×10^{0}	5.95×10^{-2}	4.23×10^{-1}
f_{14}	9.98×10^{-1}	1.99×10^{0}	1.02×10^{0}	1.40×10^{-1}
f_{15}	3.08×10^{-4}	8.08×10^{-4}	5.65×10^{-4}	1.50×10^{-4}
f_{16}	-1.03×10^{0}	-1.00×10^{0}	-1.03×10^{0}	6.52×10^{-3}
f_{17}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	1.06×10^{-6}
f_{18}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.58×10^{-14}
f_{19}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	6.57×10^{-7}
f_{20}	-3.32×10^{0}	-3.20×10^{0}	-3.30×10^{0}	3.84×10^{-2}
f_{21}	-1.02×10^{1}	-5.10×10^{0}	-1.00×10^{1}	7.22×10^{-1}
f_{22}	-1.04×10^{1}	-4.64×10^{0}	-9.99×10^{0}	1.36×10^{0}
f_{23}	-1.05×10^{1}	-5.18×10^{0}	-1.03×10^{1}	9.72×10^{-1}

表 6 和表 7 给出了这些算法对 23 个测试函数的优化结果. 其中, FEP 和 CEP 的优化结果引自文献 [20]; FES 和 CES 的优化结果引自文献 [22]; GA, PSO 和 GSO 的优化结果引自文献 [25]; RCBBO 的优化结果引自文献 [27]; DE, G3PCX 和 RCCRO

表 6 FECO 算法与其他启发式算法的比较 (1)

Table 6 Comparison between FECO and other heuristic algorithms (1)

	GA	CEP	FEP	CES	FES	DE	G3PCX	PSO
f_1	3.17×10^{0}	2.20×10^{-4}	5.70×10^{-4}	3.40×10^{-5}	2.50×10^{-4}	6.58×10^{-6}	6.40×10^{-79}	3.69×10^{-37}
算法排序	15	11	13	10	12	9	1	2
f_2	5.77×10^{-1}	2.60×10^{-3}	8.10×10^{-3}	2.10×10^{-2}	6.00×10^{-2}	2.89×10^{-4}	2.80×10^{1}	2.92×10^{-24}
算法排序	14	8	9	10	12	6	16	1
f_3	9.75×10^{3}	5.00×10^{-2}	1.60×10^{-2}	1.30×10^{-4}	1.40×10^{-3}	1.21×10^{4}	1.06×10^{-76}	1.20×10^{-3}
算法排序	15	10	9	6	8	16	1	7
f_4	7.96×10^{0}	2.00×10^{0}	3.00×10^{-1}	3.50×10^{-1}	5.50×10^{-3}	5.79×10^{0}	4.54×10^{1}	4.12×10^{-1}
算法排序	14	11	7	8	2	12	15	9
f_5	3.39×10^{2}	6.17×10^{0}	5.06×10^{0}	6.69×10^{0}	3.33×10^{1}	9.34×10^{1}	3.09×10^{0}	3.74×10^{1}
算法排序	15	3	2	4	8	14	1	9
f_6	3.70×10^{0}	5.78×10^{2}	0.00×10^{0}	4.11×10^{2}	0.00×10^{0}	0.00×10^{0}	9.46×10^{1}	1.46×10^{-1}
算法排序	13	16	1	15	1	1	14	9
f_7	1.05×10^{-1}	1.80×10^{-2}	7.60×10^{-3}	3.00×10^{-2}	1.20×10^{-2}	3.97×10^{-2}	9.80×10^{-1}	9.90×10^{-3}
算法排序	14	9	4	10	6	11	16	5
f_8	-1.26×10^{4}	-7.92×10^{3}	-1.26×10^{4}	-7.55×10^{3}	-1.26×10^{4}	-1.26×10^{4}	-2.58×10^{3}	-9.66×10^{3}
算法排序	5	10	7	11	6	2	15	9
f_9	6.51×10^{-1}	8.90×10^{1}	4.60×10^{-2}	7.08×10^{1}	1.60×10^{-1}	7.26×10^{-5}	1.74×10^{2}	2.08×10^{1}
算法排序	8	14	5	13	6	2	15	11
f_{10}	8.68×10^{-1}	9.20×10^{0}	1.80×10^{-2}	9.07×10^{0}	1.20×10^{-2}	7.14×10^{-4}	1.35×10^{1}	1.34×10^{-3}
算法排序	11	14	8	13	7	4	15	5
f_{11}	1.00×10^{0}	8.60×10^{-2}	1.60×10^{-2}	3.80×10^{-1}	3.70×10^{-2}	9.05×10^{-5}	1.13×10^{-2}	2.32×10^{-1}
算法排序	14	11	7	13	9	1	6	12
f_{12}	4.36×10^{-2}	1.76×10^{0}	9.20×10^{-6}	1.18×10^{0}	2.80×10^{-2}	1.89×10^{-7}	4.59×10^{0}	3.95×10^{-2}
算法排序	9	13	3	12	7	2	15	8
f_{13}	1.68×10^{-1}	1.40×10^{0}	1.60×10^{-4}	1.39×10^{0}	4.70×10^{-5}	9.52×10^{-7}	2.35×10^{1}	5.05×10^{-2}
算法排序	9	12	5	11	4	2	15	7
f_{14}	9.99×10^{-1}	1.66×10^{0}	1.22×10^{0}	2.16×10^{0}	1.20×10^{0}	1.58×10^{0}	1.23×10^{1}	1.02×10^{0}
算法排序	5	11	9	13	8	10	16	7
f_{15}	7.09×10^{-3}	4.70×10^{-4}	5.00×10^{-4}	1.20×10^{-3}	9.70×10^{-4}	5.37×10^{-4}	5.33×10^{-4}	3.81×10^{-4}
算法排序	16	4	5	14	13	7	6	3
f_{16}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.02×10^{0}	-4.93×10^{-1}	-1.02×10^{0}
算法排序	11	9	9	4	4	12	14	13
f_{17}	4.04×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	4.00×10^{-1}	5.56×10^{1}	4.04×10^{-1}
算法排序	13	7	7	7	7	12	16	13
f_{18}	7.50×10^{0}	3.00×10^{0}	3.02×10^{0}	3.00×10^{0}	3.00×10^{0}	3.48×10^{0}	8.67×10^{0}	3.01×10^{0}
算法排序	15	2	13	2	2	14	16	11
f_{19}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.60×10^{0}	-3.86×10^{0}
算法排序	7	9	9	9	9	8	15	13
f_{20}	-3.26×10^{0}	-3.28×10^{0}	-3.27×10^{0}	-3.24×10^{0}	-3.23×10^{0}	-3.32×10^{0}	-1.98×10^{-1}	-3.18×10^{0}
算法排序	10	7	8	11	12	4	15	13
f_{21}	-5.17×10^{0}	-6.86×10^{0}	-5.52×10^{0}	-6.96×10^{0}	-5.54×10^{0}	-8.74×10^{0}	-7.48×10^{-1}	-7.54×10^{0}
算法排序	14	8	12	7	11	4	15	5
f_{22}	-5.44×10^{0}	-8.27×10^{0}	-5.52×10^{0}	-8.31×10^{0}	-6.76×10^{0}	-9.20×10^{0}	-9.47×10^{-1}	-8.36×10^{0}
算法排序	14	8	13	7	11	5	15	6
f_{23}	-4.91×10^{0}	-9.10×10^{0}	-6.57×10^{0}	-8.50×10^{0}	-7.63×10^{0}	-9.23×10^{0}	-1.13×10^{0}	-8.94×10^{0}
算法排序	14	7	13	9	10	6	15	8
Friedman 排序	11.957	9.304	7.739	9.522	7.609	7.13	12.522	8.087
总排序	14	12	8	13	7	5	15	9

表 7　FECO 算法与其他启发式算法的比较 (2)

Table 7　Comparison between FECO and other heuristic algorithms (2)

	GSO	GSA	NFESA	RCBBO	RCCRO	GWO	WOA	FECO
f_1	1.95×10^{-8}	2.53×10^{-16}	8.10×10^{2}	1.39×10^{-3}	6.43×10^{-7}	6.59×10^{-28}	1.41×10^{-30}	3.22×10^{-23}
算法排序	7	6	16	14	8	4	3	5
f_2	3.70×10^{-5}	5.57×10^{-2}	4.07×10^{0}	7.99×10^{-2}	2.20×10^{-3}	7.18×10^{-17}	1.06×10^{-21}	3.18×10^{-16}
算法排序	5	11	15	13	7	3	2	4
f_3	5.78×10^{0}	8.97×10^{2}	1.79×10^{-6}	2.27×10^{1}	2.97×10^{-7}	3.29×10^{-6}	5.36×10^{-7}	1.47×10^{2}
算法排序	11	14	4	12	2	5	3	13
f_4	1.08×10^{-1}	7.35×10^{0}	5.11×10^{1}	3.09×10^{-2}	9.32×10^{-3}	5.61×10^{-7}	7.26×10^{-2}	4.22×10^{-1}
算法排序	6	13	16	4	3	1	5	10
f_5	4.98×10^{1}	6.75×10^{1}	1.81×10^{6}	5.54×10^{1}	2.71×10^{1}	2.68×10^{1}	2.79×10^{1}	5.29×10^{1}
算法排序	10	13	16	12	6	5	7	11
f_6	1.60×10^{-2}	2.50×10^{-16}	4.51×10^{-1}	0.00×10^{0}	0.00×10^{0}	8.17×10^{-1}	3.12×10^{0}	0.00×10^{0}
算法排序	8	7	10	1	1	11	12	1
f_7	7.38×10^{-2}	8.94×10^{-2}	3.86×10^{-1}	1.75×10^{-2}	5.41×10^{-3}	2.21×10^{-3}	1.43×10^{-3}	1.27×10^{-2}
算法排序	12	13	15	8	3	2	1	7
f_8	-1.26×10^{4}	-2.82×10^{3}	7.35×10^{3}	-1.26×10^{4}	-1.26×10^{4}	-6.12×10^{3}	-5.08×10^{3}	-1.15×10^{4}
算法排序	1	14	16	2	2	12	13	8
f_9	1.02×10^{0}	2.60×10^{1}	2.12×10^{2}	2.62×10^{-2}	9.08×10^{-4}	3.11×10^{-1}	0.00×10^{0}	1.23×10^{1}
算法排序	9	12	16	4	3	7	1	10
f_{10}	2.65×10^{-5}	6.21×10^{-2}	1.67×10^{1}	2.51×10^{-2}	1.94×10^{-3}	1.06×10^{-13}	7.40×10^{0}	1.53×10^{-12}
算法排序	3	10	16	9	6	1	12	2
f_{11}	3.08×10^{-2}	2.77×10^{1}	5.67×10^{0}	8.49×10^{-2}	1.12×10^{-2}	4.49×10^{-3}	2.89×10^{-4}	6.10×10^{-4}
算法排序	8	16	15	10	5	4	2	3
f_{12}	2.76×10^{-11}	1.80×10^{0}	4.43×10^{5}	3.28×10^{-5}	2.07×10^{-2}	5.34×10^{-2}	3.40×10^{-1}	4.07×10^{-3}
算法排序	1	14	16	4	6	10	11	5
f_{13}	4.69×10^{-5}	8.89×10^{0}	3.95×10^{6}	3.72×10^{-4}	7.05×10^{-7}	6.54×10^{-1}	1.89×10^{0}	5.95×10^{-2}
算法排序	3	14	16	6	1	10	13	8
f_{14}	9.98×10^{-1}	5.86×10^{0}	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	4.04×10^{0}	2.11×10^{0}	1.02×10^{0}
算法排序	1	15	4	3	1	14	12	6
f_{15}	3.77×10^{-4}	3.67×10^{-3}	7.89×10^{-4}	7.86×10^{-4}	5.56×10^{-4}	3.37×10^{-4}	5.72×10^{-4}	5.65×10^{-4}
算法排序	2	15	12	11	8	1	10	9
f_{16}	-1.03×10^{0}	-1.03×10^{0}	1.06×10^{0}	-1.03×10^{0}	-1.03×10^{0}	1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}
算法排序	3	1	16	7	4	15	1	8
f_{17}	3.98×10^{-1}	3.98×10^{-1}	4.24×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}
算法排序	4	1	15	11	4	3	6	2
f_{18}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.01×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}
算法排序	2	2	9	12	10	8	2	1
f_{19}	-3.86×10^{0}	-3.86×10^{0}	1.47×10^{-1}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}
算法排序	2	4	16	6	1	5	14	3
f_{20}	-3.27×10^{0}	-3.32×10^{0}	9.34×10^{-1}	-3.32×10^{0}	-3.32×10^{0}	-3.29×10^{0}	-2.98×10^{0}	-3.30×10^{0}
算法排序	9	2	16	3	1	6	14	5
f_{21}	-6.09×10^{0}	-5.96×10^{0}	6.80×10^{0}	-5.51×10^{0}	-1.01×10^{1}	-1.02×10^{1}	-7.05×10^{0}	-1.00×10^{1}
算法排序	9	10	16	13	2	1	6	3
f_{22}	-6.55×10^{0}	-9.68×10^{0}	6.85×10^{0}	-6.80×10^{0}	-1.04×10^{1}	-1.04×10^{1}	-8.18×10^{0}	-9.99×10^{0}
算法排序	12	4	16	10	2	1	9	3
f_{23}	-7.40×10^{0}	-1.05×10^{1}	6.81×10^{0}	-7.28×10^{0}	-1.05×10^{1}	-1.05×10^{1}	-9.34×10^{0}	-1.03×10^{1}
算法排序	11	1	16	12	3	2	5	4
Friedman 排序	6.043	9.217	14.043	8.13	3.87	5.696	7.13	5.696
总排序	4	11	16	10	1	2	5	2

的优化结果引自文献 [28]; GSA 和 GWO 的优化结果引自文献 [7]; WOA 的优化结果引自文献 [9]; NFESA 算法的代码来源于文献 [16] 的附录, 对所有测试函数分别进行了 51 次独立运行, 并取其平均值作为 NFESA 的优化结果, 算法参数采用该文献中的推荐参数, 即种群规模为 100, 个体的字符串长度为 12. 表 6 和表 7 中, 分别对每一个测试函数进行了算法优化结果的排序 (表中仅显示了各算法优化结果小数点后两位, 排序时按照精确优化结果进行排序), 并针对所有 23 个测试函数, 计算了各算法的 Friedman 排序结果, 表 6 和表 7 的最后一行对各算法的 Friedman 排序结果进行了进一步的排序.

从表 6 和表 7 的比较结果可以看出, FECO 算法对于低维多峰函数 $f_{14} - f_{23}$ 的优化效果表现得相对较好, 而对高维单峰函数 $f_1 - f_7$, 则相对差一些, 但是从表 2 和表 3 的比较结果中也可以看出, 当 FECO 算法的参数 p_s 和 p_m 取其他值时, 对于函数 $f_1 - f_7$ 可以得到更好的优化效果, 这主要是由各函数的不同特性造成的. 为了验证 FECO 算法的通用性, 本文针对所有测试函数, 采用统一的一组参数.

从另一方面观察, 对于每个测试函数, 获得最好优化效果的算法并不集中在某一个算法上, 而是分布在不同的算法之中, 对于 FECO 算法而言, 其在 f_6 和 f_{18} 上, 优于或并列于其他 15 个算法.

综合 23 个测试函数, 按照 Friedman 排序结果, FECO 算法在 16 个基于各种机制的启发式优化算法中, 处于并列第 2 名.

4.2 对 CEC2015 测试函数的优化结果比较

为了进一步测试 FECO 算法的优化性能以及参数适应性, 采用 CEC2015 测试函数集中 $D = 30$ 维的情况进行比较实验. 该测试函数集共有 15 个函数 ($f_{CEC15\_1} - f_{CEC15\_15}$)[29], 其中, $f_{CEC15\_1} - f_{CEC15\_2}$ 为单峰函数, $f_{CEC15\_3} - f_{CEC15\_5}$ 为多峰函数, $f_{CEC15\_6} - f_{CEC15\_8}$ 为混合函数, $f_{CEC15\_9} - f_{CEC15\_15}$ 为合成函数.

将 FECO 算法与其他 4 种启发式优化算法进行比较, 包括 CCLSHADE 算法 (基于重启动 LSHADE 的合作协同进化算法)[30]、ICMLSP 算法 (改进协方差矩阵学习及搜索优先算法)[31]、粒子群优化算法 (PSO) 和 NFESA 算法. 表 8 给出了这些算法对 15 个 CEC2015 测试函数的优化结果, 其中, CCLSHADE 算法的优化结果引自文献 [30]; ICMLSP 算法的优化结果引自文献 [31]; PSO 算法的代码及参数来自 CEC 2015 Special session & competition on real-parameter single objective optimization (learning based) 给出的算法示例 (http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Forms/AllItems.aspx); NFESA 算法的代码和参数与 4.1 中相同; FECO 算法的参数与上一组 23 个测试函数的实验一致, 对所有测试函数分别进行了 51 次独立运行, 并取其平均值作为 FECO 算法的优化结果. 所有算法的目标函数计算次数 (FEs) 按照文献 [29] 中的要求, 均为 300 000 次.

参照文献 [21] 中的对于进化算法结果评价的 wilcoxon 秩和检验方法, 表 8 中给出了 FECO 算法与其他 4 种算法的比较结果 (显著性水平取 0.05): "+" 表示 FECO 算法优于比较算法; "–" 表示 FECO 算法劣于比较算法; "∼" 表示 FECO 算法与比较算法相近似.

表 8 FECO 算法与其他启发式算法对于 CEC2015 测试函数的比较

Table 8 Comparison between FECO and other heuristic algorithms for CEC2015 functions

Function	CCLSHADE	ICMLSP	PSO	NFESA	FECO
f_{CEC15_1}	2.28×10^{-14} (−)	1.03×10^{-13} (−)	4.70×10^{6} (+)	3.87×10^{8} (+)	3.31×10^{6}
f_{CEC15_2}	5.07×10^{-14} (−)	4.05×10^{-5} (−)	5.94×10^{3} (∼)	2.62×10^{10} (+)	5.97×10^{3}
f_{CEC15_3}	2.02×10^{1} (+)	2.00×10^{1} (−)	2.09×10^{1} (+)	2.09×10^{1} (+)	2.01×10^{1}
f_{CEC15_4}	4.92×10^{0} (−)	2.31×10^{2} (+)	5.63×10^{1} (+)	3.15×10^{2} (+)	4.70×10^{1}
f_{CEC15_5}	3.00×10^{2} (−)	4.03×10^{3} (+)	2.74×10^{3} (+)	7.08×10^{3} (+)	1.81×10^{3}
f_{CEC15_6}	8.17×10^{1} (−)	1.47×10^{3} (−)	3.64×10^{5} (+)	7.72×10^{6} (+)	3.19×10^{5}
f_{CEC15_7}	1.50×10^{0} (−)	2.07×10^{1} (+)	9.90×10^{0} (∼)	1.24×10^{2} (+)	9.40×10^{0}
f_{CEC15_8}	1.58×10^{1} (−)	9.42×10^{2} (−)	1.52×10^{5} (+)	1.77×10^{6} (+)	9.43×10^{4}
f_{CEC15_9}	1.03×10^{2} (∼)	1.63×10^{2} (+)	1.03×10^{2} (∼)	2.22×10^{2} (+)	1.03×10^{2}
f_{CEC15_10}	6.06×10^{2} (−)	1.43×10^{3} (−)	1.11×10^{5} (−)	9.96×10^{6} (+)	2.93×10^{5}
f_{CEC15_11}	4.01×10^{2} (−)	1.12×10^{3} (+)	6.26×10^{2} (+)	8.82×10^{2} (+)	5.29×10^{2}
f_{CEC15_12}	1.04×10^{2} (−)	1.61×10^{2} (+)	1.17×10^{2} (+)	1.55×10^{2} (+)	1.06×10^{2}
f_{CEC15_13}	2.60×10^{-2} (−)	8.53×10^{-2} (+)	8.90×10^{-2} (+)	6.49×10^{0} (+)	2.65×10^{-2}
f_{CEC15_14}	3.28×10^{4} (−)	4.21×10^{4} (+)	3.51×10^{4} (+)	5.52×10^{4} (+)	3.36×10^{4}
f_{CEC15_15}	1.00×10^{2} (−)	1.27×10^{2} (+)	1.00×10^{2} (−)	6.28×10^{3} (+)	1.00×10^{2}

从表 8 可以看出, 与基本 PSO 算法相比, FECO 算法的优化结果在大部分函数上优于或相当于 PSO 算法; 与具有相似机制的 NFESA 算法相比, FECO 算法的优化结果在所有函数上均优于 NFESA 算法; 与改进的进化算法 ICMLSP 相比, FECO 算法的优化结果在多数函数上优于 ICMLSP 算法; 与改进的进化算法 CCLSHADE 相比, FECO 算法的优化结果只在少数函数上优于或相当于 CCLSHADE 算法.

实际上, 各种评价方法都难以衡量算法之间的绝对好坏, 但是本文的实验结果可以从一定程度上说明, FECO 算法具有一定的有效性和通用性, 可以较好地解决函数优化问题.

5 结论

本文拓展并详细分析了基于五行元素生克原理的五行环模型, 基于该模型, 实现了用于求解连续函数优化问题的五行环优化算法 (FECO), FECO 算法将所有元素划分为若干个环, 每个环中的元素按照五行环模型计算相互之间的作用力, 再根据每个元素受到的作用力来决定元素的更新方式, 经过一定次数的迭代运算后, 获得问题的最优解. 通过算法性能分析、参数比较实验, 以及与其他 17 个启发式算法的比较实验, 验证了 FECO 算法可以较好地解决连续函数优化问题.

FECO 算法作为一种新算法, 在以下方面可以有进一步改进的可能性: 1) 五行元素之间的相互作用, 除了相生、相克之外, 还有相乘、相侮等方式, 如将这些作用考虑进去, 可以对五行环模型进一步地改进或完善; 2) FECO 算法的关键环节可以有更多实现方式, 例如在种群结构方面, 可以设置多重环嵌套的结构; 在元素的更新方面, 也可以寻找更好的算子来实现; 3) FECO 算法除了解决 TSP 问题外, 还可以进一步研究, 解决更多组合优化问题; 4) 可将 FECO 算法与其他算法结合, 尝试获得更好的优化性能.

References

1 Kirkpatrick S, Gelatt C D Jr, Vecchi M P. Optimization by simulated annealing. *Science*, 1983, **220**(4598): 671−680

2 Goldberg D E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

3 Dorigo M, Stützle T. *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

4 Kennedy J, Eberhart R C. *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001.

5 Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 2007, **39**(3): 459−471

6 Yang X S. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2010, **2**(2): 78−84

7 Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer. *Advances in Engineering Software*, 2014, **69**: 46−61

8 Wang G G, Deb S, Cui Z H. *Monarch Butterfly Optimization, Neural Computing and Applications*. New York, NY, USA: Springer, 2015. 1−20

9 Mirjalili S, Lewis A. The whale optimization algorithm. *Advances in Engineering Software*, 2016, **95**: 51−67

10 Xiao Hui-Hui, Wan Chang-Xuan, Duan Yan-Ming, Tan Qian-Lin. Flower pollination algorithm based on gravity search mechanism. *Acta Automatica Sinica*, 2017, **43**(4): 576−594
(肖辉辉, 万常选, 段艳明, 谭黔林. 基于引力搜索机制的花朵授粉算法. 自动化学报, 2017, **43**(4): 576−594)

11 Dhiman G, Kumar V. Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 2017, **114**: 48−70

12 Slowik A, Kwasnicka H. Nature inspired methods and their industry applications − swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, 2018, **14**(3): 1004−1015

13 Feng You-Lan. *A Brief History of Chinese Philosophy*. Beijing: New World Press, 2004.
(冯友兰. 中国哲学简史. 北京: 新世界出版社, 2004.)

14 Tam S C, Tam H K, Tam L M, Zhang T. A new optimization method, the algorithm of changes, for Bin Packing Problem. In: Proceedings of the IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications. Changsha, China: IEEE, 2010. 994−999

15 Zhao R. Survivable topology design of hybrid fiber-VDSL access networks with a novel metaheuristic. In: Proceedings of the 5th Advanced International Conference on Telecommunications. Venice/Mestre, Italy, 2009.

16 Cui Y H, Guo R K, Guo D. A naïve five-element string algorithm. *Journal of Software*, 2009, **4**(9): 925−934

17 Punnathanam V, Kotecha P. Yin-Yang-pair Optimization: a novel lightweight optimization algorithm. *Engineering Applications of Artificial Intelligence*, 2016, **54**: 62−79

18 Liu M D. Five-elements cycle optimization algorithm for the travelling salesman problem. In: Proceedings of the 18th International Conference on Advanced Robotics (ICAR). Hong Kong, China, 2017.

19 Liu M D. Five-elements cycle optimization algorithm for solving continuous optimization problems. In: Proceedings of the IEEE 4th International Conference on Soft Computing & Machine Intelligence (ISCMI). Mauritius: IEEE, 2017. 75−79

20 Yao X, Liu Y, Lin G M. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 1999, **3**(2): 82−102

21 Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 2011, **1**(1): 3−18

22 Yao X, Liu Y. Fast evolution strategies. In: Proceedings of the 6th International Conference on Evolutionary Programming VI. Berlin: Springer, 1997. 151−162

23 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

24 Deb K, Anand A, Joshi D. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 2002, **10**(4): 371−395

25 He S, Wu Q H, Saunders J R. Group search optimizer: an optimization algorithm inspired by animal searching behavior. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(5): 973−990

26 Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. *Information Sciences*, 2009, **179**(13): 2232−2248

27 Gong W Y, Cai Z H, Ling C X, Li H. A real-coded biogeography-based optimization with mutation. *Applied Mathematics and Computation*, 2010, **216**(9): 2749−2758

28 Lam A Y S, Li V O K, Yu J J Q. Real-coded chemical reaction optimization. *IEEE Transactions on Evolutionary Computation*, 2012, **16**(3): 339−353

29 Liang J J, Qu B Y, Suganthan P N, Chen Q. Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, Singapore, 2014.

30 El-Abd M. Cooperative co-evolution using LSHADE with restarts for the CEC15 benchmarks. In: Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC). Vancouver, Canada: IEEE, 2016. 4810−4814

31 Chen L, Peng C D, Liu H L, Xie S L. An improved covariance matrix leaning and searching preference algorithm for solving CEC 2015 benchmark problems. In: Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC). Sendai, Japan: IEEE, 2015. 1041−1045

刘漫丹 华东理工大学信息科学与工程学院自动化系教授. 2000 年获得浙江大学控制理论与控制工程博士学位. 主要研究方向为工业过程建模与优化, 智能优化算法及其应用.
E-mail: liumandan@ecust.edu.cn
(**LIU Man-Dan** Professor in the Department of Automation, School of Information Science and Engineering, East China University of Science and Technology. She received her Ph. D. degree in control theory and control engineering from Zhejiang University, in 2000. Her research interest covers process modeling and optimization, intelligent optimization algorithms and applications.)