采用多模式飞行的乌鸦搜索算法*

冯爱武[a]，王一勇[a,b,c†]，付小朋[a]

(广西民族大学 a. 人工智能学院；b. 广西混杂计算与集成电路设计分析重点实验室；c. 广西高校复杂系统与智能计算重点实验室，南宁 530006)

摘　要：针对乌鸦搜索算法(CSA)的不足，提出采用多模式飞行的乌鸦搜索算法(MFCSA)。算法基于觅食能力的强弱，将群体分成觅食能力较强和较弱两个组，觅食能力较强者采用尾随跟踪当前群体最优目标策略，在群体信息指引下飞到群体当前最优位置附近开展搜索活动，增强了算法的局部开发能力；觅食能力较弱者采用观察和学习强者的觅食方法、遇到危险迅速飞离两种策略，前者可提升算法的全局探索能力，后者可保持种群的多样性。通过 15 个基准测试函数和两个工程应用问题的数值实验仿真结果表明，MFCSA 在优化精度、收敛速度等方面有更好的表现，增强了规避陷入局部最优的能力，稳定性更好。

关键词：乌鸦搜索算法；多模式飞行；工程约束问题；智能计算

中图分类号：TP301.6　　**文献标志码**：A　　**文章编号**：1001-3695(2022)06-019-1710-08

doi:10.19734/j.issn.1001-3695.2021.12.0607

Crow search algorithm using multi-mode flight

Feng Aiwu[a], Wang Yong[a,b,c†], Fu Xiaopeng[a]

(a. School of Artificial Intelligence, b. Guangxi Key Laboratory of Hybrid Computation & IC Design Analysis, c. Guangxi High School Key Laboratory of Complex Systems & Intelligent Computing, China Guangxi University for Nationalities, Nanning 530006, China)

Abstract: Aiming at the shortcomings of crow search algorithm (CSA), this paper proposed a crow search algorithm using multi-mode flight (MFCSA). Based on the strength of foraging ability, the algorithm divided the population into two groups: strong and weak foraging ability. Those with strong foraging ability adopted the strategy of trailing and tracking the optimal target of the current group, and flied to the vicinity of the current optimal position of the group under the guidance of the group information to carry out search activities, which enhanced the local exploitation ability of the algorithm. Those with weaker foraging ability adopted the two strategies of observing and learning the foraging methods of the strong, and flying away quickly when encountering danger, the former could improve the global exploration ability of the algorithm, and the latter could maintain the diversity of the population. Through the numerical experiment simulation of 15 benchmark test functions and 2 engineering application problems, the results show that the MFCSA has better performance in optimization accuracy, convergence speed, etc., enhances the ability to avoid falling into the local optimum, and has better stability.

Key words: crow search algorithm; multi-mode flight; engineering constraints problem; intelligent computing

0　引言

群智能优化算法在解决复杂优化问题时能提供可靠的解决方案，自 Holland 等人[1]提出遗传算法(GA)以来，群智能优化算法的研究越来越受到人们的重视。国内外学者在过去的三十年里先后提出了粒子群优化算法(PSO)[2]、蚁群算法(ACO)[3]、人工蜂群算法(ABC)[4]、人工鱼群算法(AFSA)[5]、蝙蝠算法(BA)[6]、狼群搜索算法(WSA)[7]、蜻蜓算法(DA)[8]、离子运动算法(IMO)[9]、乌鸦搜索算法(CSA)[10]、社会模拟优化算法(SMO)[11]等许多特点不一的群智能优化算法，这些优化算法为寻找复杂优化问题的解决方案提供了可靠的理论方法和技术指导。

乌鸦搜索算法(crow search algorithm, CSA)[10]是 Askarzadeh 于 2016 年提出的一种新的群智能随机优化算法，然而，由于 CSA 规定乌鸦个体飞行搜索方式比较单一、采用随机搜索的乌鸦个体数所占群体规模的比重偏大，从而导致算法搜索具有较大的盲目性，使 CSA 存在全局搜索能力偏弱、收敛速度慢、易陷入局部最优等不足。针对此类问题，文献[12]利用混沌方法来调整参数以提高算法的收敛速度，增强算法的寻优能力；文献[13]利用自适应参数来调整搜索步长，以提高个体的搜索能力；文献[14]利用自适应权重以及混合黄金正弦与飞蛾扑火算子等混合策略提升迭代效果；文献[15]将正弦余弦函数作为扰动参数嵌入 CSA 中，以提高算法的优化精度；文献[16]利用非劣解决定因子来确定个体是采用记忆搜索模式还是邻域搜索模式，使算法的局部搜索与全局搜索更加平衡；文献[17]利用柯西变异来保持种群多样性，利用自适应步长来增强个体的搜索能力；文献[18]利用扰动项来增强个体的局部搜索能力；文献[19]采用变因子加权学习机制和最优个体邻代维度交叉策略来调整个体搜索，以增强算法规避陷入局部最优的能力；文献[20]利用自适应参数来调整个体的搜索。然而，以上文献提出的各种改进版本 CSA 比标准 CSA 在优化精度方面有所提高，但规避陷入局部最优的能力有待增强，仍存在早熟收敛问题。针对这一问题，本文基于对现实乌鸦觅食活动形式多样性特征的模拟，提出一种新的采用多模式飞行的乌鸦搜索算法(crow search algorithm using multi-mode flight, MFCSA)：通过挖掘和利用群体信息来指导个体开展觅食(搜

收稿日期：2021-12-01；**修回日期**：2022-01-10　　**基金项目**：国家自然科学基金资助项目(61662005)；广西自然科学基金资助项目(2021JJA170094)

作者简介：冯爱武(1996-)，男，山西临汾人，硕士研究生，主要研究方向为计算智能；王勇(1963-)，男(通信作者)，广西南宁人，教授，硕导，博士，主要研究方向为计算智能(wangygxnn@sina.com)；付小朋(1996-)，男，重庆人，硕士研究生，主要研究方向为计算智能.

索)活动,进一步增强乌鸦个体的搜索效率,在保持种群多样性的同时加快算法的全局收敛精度,提升算法的全局搜索能力。通过数值实例仿真验证了本文算法的有效性和可行性。

1　乌鸦搜索算法

CSA 基于以下基本思想:乌鸦以群居的形式生活;乌鸦能记住藏身位置;乌鸦互相之间跟踪偷窃对方藏匿的食物;乌鸦保护其藏匿食物不被其他鸟类偷窃。设在 D 维搜索空间中有 N 只乌鸦,乌鸦 i 于 t 时刻的位置记为 $x_i(t)=[x_{i,1}(t),\cdots,x_{i,D}(t)]$,乌鸦 i 于 t 时刻记忆中的最好位置记为 $m_i(t)$。则 CSA 数学模型和基本步骤如下:

a)给定目标函数 $f(x)$,群体规模 N,乌鸦飞行步长 fl,最大迭代次数 t_{max},感知概率 AP。

b)初始化种群 $x_i(t)=[x_{i,1}(t),\cdots,x_{i,D}(t)]$,并记 $m_i(t)=x_i(t)$,$i=1,\cdots,N$。

c)评估每一个体的适应度值 $f(x_i(t))$,$i=1,\cdots,N$。

d)乌鸦 i 随机选择乌鸦 j,通过跟踪乌鸦 j 来发现其食物藏匿处。乌鸦 i 按式(1)更新位置。

$$x_i(t+1)=\begin{cases}x_i(t)+r_1\times fl\times(m_j(t)-x_i(t)) & r_2\geqslant AP\\ \text{a rand position} & \text{otherwise}\end{cases}\tag{1}$$

其中:r_1 和 r_2 为[0,1]中的均匀随机数,AP 为被跟踪乌鸦(乌鸦 j)的感知概率(CSA 在仿真实验中取 $AP=0.1$),$i=1,\cdots,N$。

e)检查新位置的可行性。检查每只乌鸦新位置是否可行,若新位置是可行的,则乌鸦更新其位置;否则,乌鸦则停留在当前位置且不移动。

f)评价新位置的适应度值。

g)更新记忆。乌鸦更新其记忆如下:

$$m_i(t+1)=\begin{cases}x_i(t+1) & \text{if } f(x_i(t+1)) \text{ is better than } f(m_i(t))\\ m_i(t) & \text{others}\end{cases}\tag{2}$$

若乌鸦 i 新位置适应度值优于记忆位置适应度值,则更新其记忆位置,否则不更新。

h)重复步骤 d)~g),直至满足终止条件。若满足终止条件,则依据适应度函数值将最优位置作为优化问题的最优解。

2　采用多模式飞行的乌鸦搜索算法(MFCSA)

分析 CSA 数学公式(1)和(2),式(1)表示乌鸦 i 尾随跟踪乌鸦 j(想偷窃乌鸦 j 藏匿之食物),或者采用盲目随机觅食行为;式(2)仅表示乌鸦 i 更新其记忆位置的条件,没有涉及位置的变更。换言之,CSA 规定乌鸦个体在觅食活动中只会采用跟踪搜索(偷窃别的乌鸦藏匿之食物)和随机搜索(食物)两种飞行方式。然而,现实乌鸦的生活习性除了具有 CSA 所描述的特性之外,还具有如下特征:a)乌鸦群体中个体觅食能力有强弱之分;b)强者善于尾随跟踪并伺机抢夺或偷窃他方藏匿之食物;c)弱者善于跟踪观察和学习强者的觅食方法,遇到危险时会迅速飞离当前位置;d)乌鸦可动态改变飞行方式。基于此,结合 CSA 数学模型,本文提出采用多模式飞行的乌鸦搜索算法。

设在 D 维搜索空间中有 N 只乌鸦,记 $x_i(t)=[x_{i,1}(t),\cdots,x_{i,D}(t)]$ 为乌鸦 i 于 t 时刻的位置,记 $m_i(t)$ 为乌鸦 i 于 t 时刻记忆中的最好位置。

2.1　群体分为觅食能力较强组与较弱组

基于乌鸦群体中不同个体的觅食(搜索)能力有强弱之分。因此,首先依据乌鸦个体觅食(搜索)能力的强弱将群体分成觅食(搜索)能力较强组和较弱组。具体方法如下:a)将乌鸦群体按个体觅食能力的强弱(依据适应度值)排序,记乌鸦 i 于 t 时刻的觅食能力在群体中排第 $o_i(t)$ 位,$i=1,\cdots,N$,例如 $o_i(t)=1$ 表示乌鸦 i 于 t 时刻的觅食能力在种群中排第 1 位(表示乌鸦 i 于 t 时刻的觅食能力最强,即 $x_i(t)$ 的适应度值最优),$o_i(t)=N$ 表示乌鸦 i 于 t 时刻在种群中排第 N 位(表示乌鸦 i 于 t 时刻的觅食能力最弱,即 $x_i(t)$ 的适应度值最差);b)依觅食能力的强弱将种群分成两个组,将排在群体前 N_1 位的 N_1 只乌鸦组成觅食能力较强组,其余 $N-N_1$ 只乌鸦组成觅食能力较弱组。记

$$\alpha_i(t)=(o_i(t)-1)/(N-1)\tag{3}$$

$$AF_{i,j}=1/(1+\exp(-0.1/d(i,j)))\tag{4}$$

本文分别称 $\alpha_i(t)$ 和 AF_i 为乌鸦 i 于 t 时刻的步长因子和多模式步长。其中 $d(i,j)=|x_{i,j}(t)-m_{best,j}(t)|$,表示第 i 只乌鸦的第 j 维与最优者记忆位置的第 j 维的距离($j=1,\cdots,D$)。式(3)表示若乌鸦 i 于 t 时刻的觅食能力最强,则 $o_i(t)=1$,从而 $\alpha_i(t)=0$;若乌鸦 i 于 t 时刻的觅食能力最弱,则 $o_i(t)=N$,从而 $\alpha_i(t)=1$。故觅食能力越强者,其步长因子越小。这也是为了让适应度较好的乌鸦配备更小的步长,防止步长过大遗漏更优值。

2.2　不同组别的个体采用不同的搜索策略

a)强者善于尾随跟踪并伺机抢夺或偷窃他方藏匿的食物。对于 $x_i(t)\in N_1$,设计其按如下公式更新位置:

$$x_{i,j}(t+1)=\begin{cases}x_{i,j}(t)+r_1\times\alpha_i(t)\times AF_{i,j}\times(m_{best,j}(t)-x_{i,j}(t)) & \text{if } r_2\geqslant AP_1\\ m_{best,j}(t)+\delta & \text{else}\end{cases}\tag{5}$$

其中:$m_{best}(t)=[m_{best,1}(t),\cdots,m_{best,D}(t)]$ 表示至 t 时刻 N_1 中觅食能力最强者的记忆位置,r_1,r_2 均为(0,1]中的随机数,δ 为($-\varepsilon,\varepsilon$)内的随机数,ε 是一个比较小的正函数。本文在仿真实验中取 AP_1 为(0,0.1)中的随机数,$\varepsilon=(10t)^{-3}$。

式(5)表示:(a)能力较强组 N_1 中的个体搜索步长由 $\alpha_i(t)$ 与 AF_i 共同控制,相比 CSA 固定的飞行步长,此步长结合了位置间距离因素和适应度因素,使得较强组中的乌鸦能更适应地飞到优质区域展开局部搜索,尤其是在迭代后期,避免了因固定的飞行步长而导致位置来回扰动,使搜索迟迟不能收敛到全局最优位置;(b)若 $r_2\geqslant AP_1$,$x_{i,j}(t)+r_1\times\alpha_i(t)\times AF_{i,j}\times(m_{best,j}(t)-x_{i,j}(t))$ 表示乌鸦 i(位于 $x_i(t)$)正在飞往最优记忆位置 $m_{best}(t)$ 的途中搜索;否则,$m_{best,j}(t)+\delta$ 则表示乌鸦 i 已经飞到 $m_{best}(t)$ 的附近展开搜索。采用这样的搜索策略,使得觅食能力较强者专注于优质局部区域的深度开发,提升了个体的搜索效率和搜索精度。

b)觅食能力较弱者倾向于观察和学习强者的觅食方法,遇到危险则迅速飞离。因此,对于 $x_i(t)\in N-N_1$,设计其按如下公式更新位置:

$$x_{i,j}(t+1)=\begin{cases}wx_{i,j}(t)+r_3\times fl\times((m_{q1,j}(t)-x_{i,j}(t))-\\ \quad(m_{q2,j}(t)-x_{i,j}(t))) & \text{if } r_4\geqslant AP_2\\ x_{i,j}(t)+r_5\times fl\times v_i(t) & \text{else}\end{cases}\tag{6}$$

其中:w 被称为维度决策因子,当 $D\leqslant 5$ 时(D 为搜索空间维数),取 $w=1$,而当 $D>5$ 时,取 $w=(t_{max}-t)/t_{max}$。该处理可以根据搜索空间维度合理调控对自身位置学习的比重,在高维优化问题中,随着迭代次数的增加可自动缩小搜索范围,加快了算法的收敛速度。$m_{q1}(t)$ 和 $m_{q2}(t)$ 是从觅食能力较强组 N_1 中随机选取的两只乌鸦;r_3、r_4 均为(0,1]中的随机数,r_5 为[-1,1]中的随机数;$v_i(t)=\dfrac{|f(m_{best}(t))|}{1+|f(x_i(t))|}$,本文称 $v_i(t)$ 为乌鸦 i 于 t 时刻的逃离速度绝对值。本文在仿真实验中取 $AP_2=0.1$。

式(6)表示当 $r_4\geqslant AP_2$ 时,乌鸦 i 采用观察和学习强者搜索策略,$wx_{i,j}(t)+r_3\times fl\times((m_{q1,j}(t)-x_{i,j}(t))-(m_{q2,j}(t)-x_{i,j}(t)))$ 表示乌鸦 i 观察和学习强者 $m_{q1}(t)$ 和 $m_{q2}(t)$ 的觅食方法,其融合了差分进化算法思想,随机选择了觅食较强组的

两只乌鸦生成差分向量(如($m_{q1,j}(t) - x_{i,j}(t)$就是一个差分向量),最后与个体向量相加生成新的子代位置向量。乌鸦受两只乌鸦的共同吸引,利用位置差异信息可以减少种群追逐单一乌鸦后都陷入局部极值的可能性,扩大搜索范围;选择觅食能力较强者的目的是利用较优位置信息来指导个体开展觅食搜索活动,使得逼近最优位置的效果更加显著。因此,算法在迭代前期,种群位置差异较大,可使算法有较强的全局搜索能力;在搜索后期,种群位置差异相对较小,算法仍可保留较强的局部搜索能力和稳定性。当 $r_4 < AP_2$ 时,$x_{i,j}(t) + r_5 \times fl \times v_i(t)$ 表示乌鸦 i 遇到危险时快速逃离当前位置。$v_i(t)$ 表示位于 $x_i(t)$ 的乌鸦 i 的适应度越好(即 $f(x_i(t))$ 越小),其逃离速度绝对值就越大,搜索范围相应也就越大。换言之,由于种群中并不缺乏适应度相对更好的个体(如觅食能力较强组中的个体),为了防止所有乌鸦都朝着局部最优的位置飞行,导致乌鸦集群化,致使一些区域未得到充分挖掘,需要少部分个体增加扰动项。这部分个体是从适应度相对较好(但不是最好,最好的在能力较强组)的个体中选取的,因为其中有些可能已经陷入了局部最优,这样使之具有跳出局部最优的能力,同时也避免了标准 CSA 为了增加全局搜索能力而选择盲目的随机落位,可能会导致原有最优解没有得以继承,致使收敛速度较慢。

综上所述,觅食能力较强者专注于优秀成员的局部搜索,使全局最优位置信息在较优个体间得到及时反馈,在此基础上将其附近的微小邻域完全开发,体现了种群最优区域下的纵深挖掘能力。觅食能力较弱者相较于前者全局搜索能力得到了一定的提升,利用位置差信息来指引搜索方向,拓展了搜索范围,加快了搜索的收敛速度;选择一少部分可能已陷入“假最优”的较优个体作逃逸处理,提升了全局探索能力,其产生的优秀子代也为能力较强者下一次进行的局部搜索创造了条件。

觅食能力较强组与较弱组的合理划分可平衡全局勘探与局部开发,特别是对于较为复杂的优化问题,有利于减少陷入局部最优的可能性,多样性种群的分工和协作有利于个体各司其职,在搜索效率与搜索精度得到提升的同时,兼顾了更好的稳定性。

2.3 MFCSA 流程

本文算法(MFCSA)流程如下:a)给定搜索空间维数 D,目标函数 $f(x)$,种群规模 N,飞行步长 fl,最大迭代次数 t_{max},感知概率 AP_2;b)初始化种群 $x_i(t)$,并记 $m_i(t) = x_i(t)$,$i = 1, \cdots, N$;c)评价每一个体的适应度值 $f(x_i(t))$,$i = 1, \cdots, N$;d)依据适应度值将种群排序,将排在前面的 N_1 只乌鸦作为觅食能力较强组,其余的 $N - N_1$ 只乌鸦作为较弱组,并按式(3)(4)计算 $\alpha_i(t)$,AF_i,$i = 1, \cdots, N$;e)觅食能力较强组按式(5)更新位置,觅食能力较弱组按式(6)更新位置;f)评价每一个体新位置的适应度值;g)按式(2)更新每一个体的记忆;h)重复步骤 d)~g),直至满足终止条件。若满足终止条件,则依据适应度值将最优位置作为优化问题的最优解。

3 数值实验及结果分析

3.1 仿真环境

该实验使用 MATLAB R2020a 进行仿真,操作系统为 Windows 10,电脑配置 8 GB 内存,处理器为 Intel® Core™ i5-6300HQ CPU@2.30 GHz。

3.2 测试函数选取

选取 15 个比较典型的基准函数(表 1)作为本次算法性能测试分析(都是求最小值问题),其中 $F_1 \sim F_5$ 为高维单峰函数,F_{14} 为高维阶梯函数,$F_6 \sim F_{13}$ 为高维多峰函数,F_{14} 为 2 维单峰函数,F_{15} 为 2 维多峰函数。

表 1 基准测试函数

Tab. 1 Benchmark functions

基准测试函数	类型	维度 D	搜索空间	最优值
$F_1(x) = \sum_{i=1}^{D} x_i^2$	单峰	30	$[-100,100]$	0
$F_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	单峰	30	$[-10,10]$	0
$F_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	单峰	30	$[-100,100]$	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant D \}$	单峰	30	$[-100,100]$	0
$F_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	单峰	30	$[-30,30]$	0
$F_6(x) = -\sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	多峰	30	$[-500,500]$	$-418.982D = -12569.46$
$F_7(x) = 10D + \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i)]$	多峰	30	$[-5.12,5.12]$	0
$F_8(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	多峰	30	$[-32,32]$	0
$F_9(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	多峰	30	$[-600,600]$	0
$F_{10}(x) = \frac{\pi}{D}\{10\sin(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} \mu(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$, $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	多峰	30	$[-50,50]$	0
$F_{11}(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	多峰	30	$[-10,10]$	0
$F_{12}(x) = (\sum_{i=1}^{D} \lvert x_i \rvert) / \exp(\sum_{i=1}^{D} \sin(x_i^2))$	多峰	30	$[-2\pi,2\pi]$	0
$F_{13}(x) = \frac{1}{D}\sum_{i=1}^{D} \sin(10\log(x_i))$	多峰	30	$[0.25,10]$	-1
$F_{14}(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	单峰	2	$[-1.2,1.2]$	0
$F_{15}(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2 [1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2 [1 + \sin^2(2\pi x_2)]$	多峰	2	$[-10,10]$	0

3.3 探究乌鸦类别比例对 MFCSA 性能的影响

将整个种群划分成两个类别,合适的比例划分可以使算法达到勘探与开发的平衡,在求解问题时能兼备效率和精度。因此研究更合适的划分比例,即转换为探究觅食较强者 N_1 所占种群 N 的比例问题,分别取占种群规模 N 为 10%、20%、30%、50%、80% 进行对比实验,其中 $fl = 2$,$AP_2 = 0.1$,每一比例的实验均独立运行 30 次,种群规模 $N = 30$,搜索维度 $D = 30$,迭代次数 $t_{max} = 500$。

测试函数选择 F_6,其是一个具有单一最小值的高维多峰函数,最小值为 $-418.982D = -12569.46$,因为具有很多极小值,且在(420.9687,…,420.9687)取得最优值,优化条件要求较高,很多优化算法都难以找到其全局最优值。该测试函数比较能考验算法的全局搜索性能和局部纵深挖掘能力。

此外,还要重点考查算法的鲁棒性,即每次独立运行结果数据分布情况,稳定的数据分布对工程问题的求解提供了可行性。因而通过绘制箱线图来刻画 30 次运行结果,可反映多组

连续型定量数据分布的中心位置和散布范围。

由图 1 可知,五种比例下 MFCSA 中位数线(红色横线,见电子版)均靠近全局最小值,对于半数以上数据都是接近理论值的,这说明此算法本身有较强的逃逸局部极值能力和优秀的收敛精度,算法各个策略与整体机能是有效和可行的。此外,从箱线图的高度来看,觅食较强组占比 30% 与 50% 时数据分布是最稳定的,大部分数据都有着相接近且较高的精度;再从离群值(红色加号)观察,占比 50% 的离群值比占比 30% 相对更大,其次由于较弱组全局搜索和局部搜索(局部搜索不如较强组)并存,其搜索到的优秀位置也可能成为较强组继续展开搜索的条件,所以较强组的比例不应该太高,觅食较强组占比 30% 是一个不错的选择,可显著地改善了算法的性能。

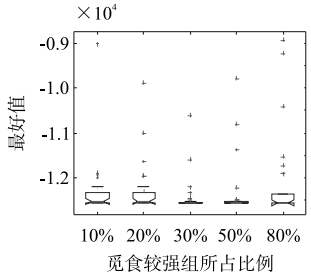

图 1　不同比例划分下 MFCSA 表现

Fig. 1　Performance of MFCSA under different proportion division

3.4　算法对比与参数设置

为了验证 MFCSA 的性能,将 MFCSA 与标准 CSA[10]、PSO[2] 以及最近发表的改进版乌鸦搜索算法[12~20] 中选取了两个优化能力比较强的 SCA-CSA[15] 和 ICSA[19] 进行性能对比分析。实验中,所有对比算法的种群规模 $N=30$,迭代次数 $t_{max}=500$;其余参数与对应参考文献保持一致以保证公平性。五种算法主要参数设置如表 2 所示。

表 2　重要参数设置

Tab. 2　Important parameter setting

算法	参数设置
CSA	$fl=2, AP=0.1$
SCACSA	$fl=2, AP=0.1$
ICSA	$fl\in[1.5,2.5], AP\in[0.05,0.15], \lambda\in[0.05,0.95]$ $AP_{CvDF}=AP_{max}\times\exp(\log(\frac{AP_{min}}{AP_{max}})\times\frac{t}{t_{max}})fl_{CvDF}=fl_{max}\times$ $\exp(\log(\frac{fl_{min}}{fl_{max}})\times\frac{t}{t_{max}})\lambda_{CvDF}=\lambda_{max}\times\exp(\log(\frac{\lambda_{min}}{\lambda_{max}})\times\frac{t}{t_{max}})$
PSO	$c1=2, c2=2, w=0.9$
MFCSA	$fl=2, AP_2=0.1, N_1=N\times30\%$ $w=\begin{cases}1 & \text{if } D\leqslant5\\ (t_{max}-t)/t_{max} & \text{if } D>5\end{cases}$

3.5　对比结果分析

为了避免随机性对测试结果的影响,本文做数值实验测试时,每一种算法针对每一测试函数都独立运行 30 次,并将每次运行得到的最优值、平均值、标准差记录下来。这些数据指标总体上反映了算法优化能力的强弱,其中,最优值能够体现算法的寻优精度,平均值和标准差能够体现算法的稳定性能,得到的实验结果如表 3 所示。

从最优值指标上看:a)对于高维单峰函数 $F_1\sim F_5$,只有 MFCSA 能找到 F_1、F_2、F_3、F_4 这四个函数的理论最优值,由此可知精度至少领先标准 CSA 150 ~ 300 个数量级,排名第二的 SCACSA 在 F_1 这个最简单的单峰函数上也只不过达到了 1.8E -60,具有香蕉形状的 F_5 由于靠近极小值位置需要经历较大的弯路,比较影响算法收敛速度和精度,很容易陷入停滞点,MFCSA 找到 F_5 的最优值达到了 4E -2,领先第二名 ICSA 超过 2 个数量级;b)对于 2 维单峰函数 F_{14},MFCSA 找到 F_{14} 的最优值比其他四种算法更接近理论最优值;c)对于高维多峰函数 $F_6\sim F_{13}$,MFCSA 能找到 F_6、F_7、F_9、$F_{11}\sim F_{13}$ 这六个函数的理论最优值,SCACSA 只找到了 F_7、F_9 的理论最优值,其余算法均没有找到这八个函数的理论最优值,除本文算法外,其余算法对于 F_6、F_{13} 这两个存在强烈振荡的测试函数都很难寻到“非 0 最优值”,意味着存在早熟收敛现象,MFCSA 找到 F_8、F_{10} 的最优值比 CSA 分别领先超过 15 和 3 个数量级;d)对于 2 维多峰函数,MFCSA 找到 F_{15} 的最优值比 CSA 超过 10 个数量级。MFCSA 在所有测试函数中均位列第一,说明不论是针对单峰还是多峰优化问题,特别是局部最优束缚能力较强的振荡函数,MFCSA 均比其他四种算法有不错的全局搜索能力和显著优化精度。

从平均值和标准差指标上看,在对 15 个基准函数的数值实验测试中,MFCSA 相较于其他四种算法始终位列第一,说明其鲁棒性和稳定性比较好,为解决未知最优解的工程优化问题提供了可靠性。

为了更直观地对比本文五种算法各自的收敛速度,给出五种算法在求解八个有代表性的测试函数的适应度曲线对比,包括 F_1、F_3、$F_5\sim F_7$、F_{10}、F_{14}、F_{15},结果如图 2 所示。

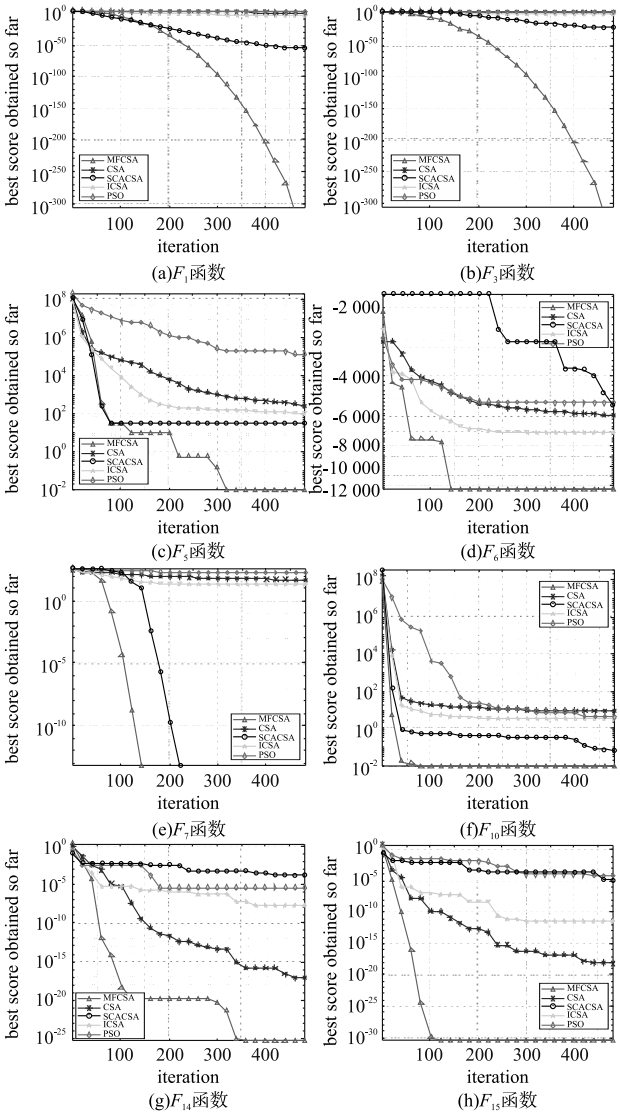

图 2　八个函数收敛曲线

Fig. 2　Convergence curves of 8 functions

从图 2 可以看出,MFCSA 的收敛曲线相较于其他四种算法下降的速度最快、并且均位于其他四种算法的收敛曲线图的

最下方位置;对于 F_1、F_3、F_7 这三个测试函数,MFCSA 的收敛曲线均达到理论最优位置,而 SCACSA 只有 F_7 的收敛曲线到达理论最优位置。

以相对简单的测试函数 F_1、F_3 为例,主要考验的是算法的收敛速度,MFCSA 在前期的梯度绝对值相对较小,说明其在前期尽可能扩大搜索范围,在后期下降非常快,表明加强了局部搜索,能快速找到最优位置。其他算法如 CSA 由于整体偏于勘探,几乎处于停滞状态;对于 F_5、F_6、F_{14} 可以发现本文算法出现了几个间断式跳跃的拐点,与此同时,CSA、SCACSA、ICSA、PSO 几乎全陷入了局部最优点,出现了早熟收敛的状态。这说明 MFCSA 跳出局部最优的能力是最强的;针对 F_7、F_{10}、F_{15} 的测试,MFCSA 平均不到 150 次就达到了收敛,说明收敛速度比较快。

表 3 实验结果对比

Tab. 3 Comparison of experimental results

函数	算法	最优值	平均值	标准差	函数	算法	最优值	平均值	标准差
	MFCSA	0	0	0		MFCSA	0	0	0
	CSA	2.387192	7.32535	2.540898		CSA	0.994006	1.071012	0.040107
F_1	SCACSA	1.869347E-60	8.12396E-54	4.320094E-53	F_9	SCACSA	0	2.611406E-04	0.00143
	ICSA	1.23327E-05	1.180691E-04	1.166355E-04		ICSA	6.53399E-04	0.022152	0.031547
	PSO	94.05799	3.419035E+02	1.743048E+02		PSO	2.13884	4.411911	1.992826
	MFCSA	0	0	0		MFCSA	5.797166E-04	0.011463	0.008461
	CSA	1.760479	3.299293	0.905106		CSA	1.779838	4.859254	1.755932
F_2	SCACSA	1.052618E-37	1.822705E-31	9.898236E-31	F_{10}	SCACSA	0.027556	0.065334	0.017755
	ICSA	0.023091	0.071432	0.035114		ICSA	0.005361	2.577068	1.536027
	PSO	17.4564	46.95162	22.31221		PSO	2.300614	6.778687	3.402027
	MFCSA	0	0	0		MFCSA	0	0	0
	CSA	1.401275E+02	3.315817E+02	99.29079		CSA	0.063782	1.460956	1.128487
F_3	SCACSA	5.819498E-24	6.499796E-09	3.515372E-08	F_{11}	SCACSA	3.682216E-36	6.730493E-28	2.561222E-27
	ICSA	0.101634	0.297717	0.197661		ICSA	0.015058	1.490089	1.665236
	PSO	3.976962E+03	1.576034E+04	1.341208E+04		PSO	5.626784	16.34764	5.494502
	MFCSA	0	0	0		MFCSA	0	0	0
	CSA	3.558967	5.737388	1.237143		CSA	5.626665E-12	3.011937E-11	2.698268E-11
F_4	SCACSA	2.699932E-17	9.825602E-09	4.742352E-08	F_{12}	SCACSA	2.979666E-09	1.70752E-07	2.663446E-07
	ICSA	0.076127	0.18603	0.080384		ICSA	7.870121E-12	1.627641E-11	2.73713E-12
	PSO	3.556379	9.50948	3.129581		PSO	9.42178E-12	2.9929450E-11	5.705435E-12
	MFCSA	0.043278	1.247388	0.377937		MFCSA	-1	-1	2.931387E-05
	CSA	1.312759E+02	4.178955E+02	2.402964E+02		CSA	-0.795622	-0.626829	0.072298
F_5	SCACSA	26.73717	27.55369	0.558041	F_{13}	SCACSA	-0.996776	-0.881358	0.148455
	ICSA	25.36709	30.39785	11.20197		ICSA	-0.826881	-0.684003	0.082819
	PSO	3.305925E+03	2.630415E+04	2.114117E+04		PSO	-0.705656	-0.500373	0.092403
	MFCSA	-1.2569487+04	-1.253157E+04	71.17916		MFCSA	3.752019E-29	4.437828E-18	1.248822E-17
	CSA	-7.8832102E+03	-6.7608023E+03	6.47309E+02		CSA	1.440016E-20	3.721411E-17	1.56488E-16
F_6	SCACSA	-7.635352E+03	-5.146446E+03	1.11633E+03	F_{14}	SCACSA	2.630476E-07	2.676076E-05	4.684164E-05
	ICSA	-1.038306E+04	-7.375125E+03	1.308059E+03		ICSA	9.744849E-10	7.722976E-08	1.052358E-07
	PSO	-9.2822276E+03	-7.428961E+03	8.772826E+02		PSO	5.575874E-08	3.951818E-04	0.001196
	MFCSA	0	0	0		MFCSA	1.349783E-31	3.019098E-19	1.652128E-18
	CSA	14.50187	30.21445	9.933084		CSA	1.098546E-20	4.786611E-18	8.042859E-18
F_7	SCACSA	0	0	0	F_{15}	SCACSA	6.374604E-08	3.174027E-06	3.521644E-06
	ICSA	12.1727	31.09064	12.96691		ICSA	2.374131E-12	8.298609E-10	1.587751E-09
	PSO	1.349089E+02	2.358207E+02	37.160203		PSO	2.486005E-06	2.353222E-04	2.336836E-04
	MFCSA	8.881784E-16	8.881784E-16	0					
	CSA	2.171871	3.843137	0.87049					
F_8	SCACSA	4.440892E-15	5.033011E-15	1.346653E-15					
	ICSA	1.795722	3.879881	1.106917					
	PSO	4.813672	10.40343	5.600888					

为研究 MFCSA 在不同迭代阶段的种群分布状况,选择了 2 维多峰测试函数 F_{15},此函数在(1,1)处取得全局最优值 0。种群在迭代 500 代的过程中,记录了第 1、10、20、100 代的种群分布图,并与标准 CSA 进行了对比,结果如图 3 ~ 6 所示。

由图 3 可观察到两个算法开始时都趋于比较相似的分布,以证明公平性;图 4 展示了至第 10 代,MFCSA 种群中重合点较多,说明大部分个体已接近或到达最优位置,而标准 CSA 种群中的大部分个体仍处于聚集过程中,相对来说,距最优位置仍有一段距离,表明 MFCSA 种群之间的信息交流更快更有效;图 5 说明 MFCSA 到第 20 代时,种群中个体几乎都到达了理论最优位置,而 CSA 种群中大部分个体只到达理论最优位置的附近,MFCSA 明显领先于 CSA;图 6 中,MFCSA 出现了离开理论位置的点,这是因为其存在逃逸机制,大多数情况下都不会使种群所有个体完全聚在一个位置上,而 CSA 基本所有个体都被吸引到同一个位置上,若这一位置不是全局最优,算法搜索也就无法摆脱当前局部最优位置而陷入局部极值。结合图 2(h)的收敛曲线来看,在第 100 代, MFCSA 最优值领先 CSA 至少 20 个数量级,说明 MFCSA 一方面利用觅食较强组找到最优值的个体信息继续增加开发深度,这便是领先 CSA 的原因;另一方面,觅食较弱组的部分个体通过危险逃离机制不断扰动,以此来提升一部分全局搜索能力,再次表明 MFCSA 有较强的逃逸局部极值的能力。

基于此,相较于 CSA、SCACSA、ICSA、PSO 四种算法,MFCSA 具备最好的整体寻优与局部寻优的协同能力,因此多样性种群分工合作是可行的,有利于防止因局部极值导致早熟收敛的问题,有利于提升收敛速度及精度。

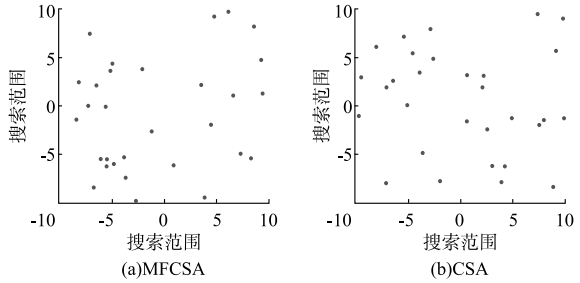

图 3　种群第 1 代分布图

Fig. 3　Distribution map of the 1st generation of population

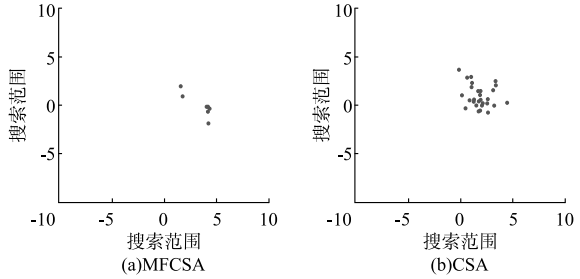

图 4　种群第 10 代分布图

Fig. 4　Distribution map of the 10th generation of population

3.6　求解不同规模优化问题比较

为了进一步测试五种算法的性能,本文再对表 1 中的 F_1、F_3、$F_5 \sim F_7$ 这五个基准测试函数分别取 $D=50,100,200$ 时,测试这五种算法的优化性能。得到的数值实验结果如表 4 所示。从表 4 中最优值指标上看,对于 50、100、200 维三种不同情况,MFCSA 均能从五个测试函数中找到 F_1、F_3、F_7 这三个函数的理论最优值,且寻优结果变化不大。其他四种对比算法中,只有 SCACSA 找到 F_7(50 维)的理论最优值,其余三种算法均没有找到这五个测试函数的理论最优值。MFCSA 找到 F_5、F_6 的最优值已经非常接近理论最优值,且比其他四种算法都好,特别是对于 F_5,CSA、SCACSA、ICSA、PSO 这四种算法的寻优结果出现了“失灵”现象,找到的最优值与理论最优值相差甚远。基于此,相较于其他四种算法,MFCSA 的全局搜索能力更强,在解决不同规模优化问题时均具有较好的收敛速度和优化精度。

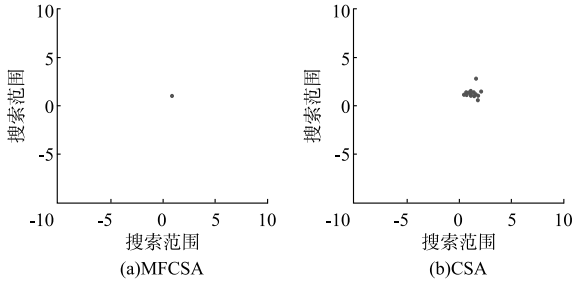

图 5　种群第 20 代分布图

Fig. 5　Distribution map of the 20th generation of population

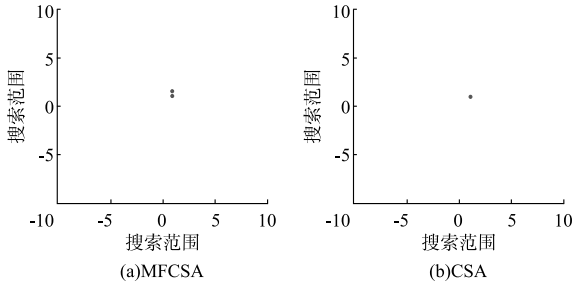

图 6　种群第 100 代分布图

Fig. 6　Distribution map of 100th generation of population

表 4　不同维度实验结果对比

Tab. 4　Comparison of experimental results with different dimensions

函数	算法	50 dim			100 dim			200 dim		
		最优值	平均值	标准差	最优值	平均值	标准差	最优值	平均值	标准差
F_1	MFCSA	0	2.642419E-142	1.446571E-141	0	1.176299E-136	6.442857E-136	0	3.33929E-140	1.826569E-139
	CSA	46.925259	92.66940	24.54534	5.322687E+02	7.396559E+02	1.071902E+02	2.242612E+03	2.835995E+03	2.561688E+02
	SCACSA	6.257444E-46	1.122879E-31	3.564858E-31	4.819588E-14	2.061783E-06	8.070327E-06	0.15776	19.12498	29.08524
	ICSA	0.006919	0.033589	0.019652	0.778518	2.115455	0.558564	15.10255	21.93689	3.754682
	PSO	94.67256	1.638742E+03	1.858208E+03	1.608155E+03	7.382385E+03	4.961644E+03	8.812701E+03	2.024944E+04	8.539767E+03
F_3	MFCSA	0	1.199172E-147	6.568137E-147	0	9.794424E-158	5.364625E-157	0	4.56836E-156	2.411236E-155
	CSA	7.510436E+02	1.28454E+03	2.736512E+02	4.180063E+03	6.234765E+03	1.128117E+03	1.668612E+04	2.534714E+04	5.026036E+03
	SCACSA	2.526163E-09	4.779111	10.86633	1.666821E+02	1.269622E+04	9.501924E+03	9.706401E+04	1.719494E+05	7.166905E+04
	ICSA	0.411708	1.578723	0.593039	2.182817	10.46192	4.593752	7.553917	37.28681	17.79766
	PSO	1.526526E+04	4.456862E+04	2.455347E+04	7.873876E+04	1.698723E+05	5.504416E+04	3.045116E+05	6.351465E+05	2.701043E+05
F_5	MFCSA	0.001592	2.021069	2.775918	0.001513	3.434317	5.323429	0.022637	11.36925	16.00047
	CSA	1.030471E+03	2.402563E+03	9.246995E+02	1.594998E+04	2.285951E+04	5.076837E+03	7.832656E+04	1.256569E+05	3.101487E+04
	SCACSA	46.99322	48.16803	3.493527	97.10629	98.37295	5.459131	2.341277E+02	2.150621E+04	4.210774E+04
	ICSA	48.24428	50.26514	2.50237	1.197172E+02	1.371906E+02	13.73998	3.226034E+02	3.977496E+02	43.68478
	PSO	2.004942E+04	2.770973E+06	1.461899E+07	3.282791E+05	1.4592E+07	3.038668E+07	1.238436E+06	8.001524E+07	1.011186E+08
F_6	MFCSA	-2.094914E+04	-2.083972E+04	2.278102E+02	-4.189828E+04	-4.158815E+04	1.177863E+03	-8.379657E+04	-8.229695E+04	1.349917E+03
	CSA	-1.116664E+04	-9.622298E+03	8.009292E+02	-1.696706E+04	-1.461604E+04	1.458359E+03	-2.620690E+04	-2.122588E+04	2.094456E+03
	SCACSA	-1.005143E+04	-8.387086E+03	1.100666E+03	-1.658069E+04	-1.314723E+04	1.675686E+03	-2.145266E+04	-1.743519E+04	2.594886E+03
	ICSA	-1.537513E+04	-1.134116E+04	1.914771E+03	-2.578309E+04	-1.829909E+04	4.868611E+03	-4.049602E+04	-2.950491E+04	6.554422E+03
	PSO	-1.253746E+04	-1.003333E+04	1.409698E+03	-2.079266E+04	-1.578949E+04	2.620857E+03	-2.527847E+04	-2.142951E+04	2.344332E+03
F_7	MFCSA	0	0	0	0	0	0	0	0	0
	CSA	52.22839	83.34671	23.82069	2.116018E+02	3.120477E+02	41.30439	8.926953E+02	9.762317E+02	52.05811
	SCACSA	0	1.597451	5.492739	3.589093E-10	1.825009E+02	1.646642E+02	4.853826E+02	1.277529E+03	2.998594E+02
	ICSA	23.43424	44.54704	14.26252	89.49353	1.63714E+02	34.05817	5.219452E+02	6.382015E+02	77.70969
	PSO	3.043432E+02	4.242599E+02	51.28564	5.418741E+02	9.354586E+02	9.354586E+02	1.420693E+03	1.965518E+03	2.186017E+02

从平均值、标准差指标上看,MFCSA 求解 F_7 时,随着维数从 50 增加到 200,找到的平均值、标准差都与理论最优值相同,其他四种算法找到的平均值、标准差都呈现逐步偏离理论最优值现象;MFCSA 求解 F_1、F_3、F_6 时,随着维数从 50 增加到 200,找到的平均值、标准差都非常接近,甚至有些高维结果相对更好,其他四种算法找到的平均值、标准差均呈现逐步偏离

理论最优值现象;五种算法在求解 F_5 时,随着维数增加,找到的平均值、标准差均呈现逐步偏离理论最优值现象,但 MFCSA 的表现是最好的。基于以上分析,相较于其他四种算法,MFCSA 能有效避免“维灾难”,规避陷入局部最优的能力最强,算法的稳定性最好。

4 MFCSA 在实际工程中的应用

工程应用的约束优化是一类有实际意义的数学规划问题,其约束机制使得搜索空间可行域的分布减少,这为优化问题带来了挑战。本文将 MFCSA 应用于解决二个工程设计问题,并与多种算法求解结果进行对比,通过收敛曲线对比图来说明 MFCSA 可用于解决具体工程问题。

4.1 三杆桁架设计问题

三杆桁架设计的目的是通过调整横截面积(x_1,x_2),使受应力(σ)约束的三杆桁架的体积最小。图 7 为设计示意图。该优化问题有如下三个非线性不等式约束:

$$\min \quad f(x)=(2\sqrt{2}x_1+x_2)\times l$$

$$\text{s.t.} \quad g_1(x)=\frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0$$

$$g_2(x)=\frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0$$

$$g_3(x)=\frac{1}{\sqrt{2}x_2+x_1}P-\sigma\leqslant 0$$

$$0\leqslant x_i\leqslant 1,i=1,2$$

$$l=100\ \text{cm},P=2\ \text{kN/cm}^2,\sigma=2\ \text{kN/cm}^2 \qquad (7)$$

本次实验设置迭代次数是 500 次,对比算法各独立运行 30 次,其余参数设置与相关参考文献保持一致。MFCSA 求解三杆桁架设计问题时得到的最优解决方案如表 5 所示,与 CSA[10]、MVO[21]、GOA[22]、MFO[23]、WCA[24]、MBA[25] 对比结果如表 6 所示;图 8 是 MFCSA 求解该问题的进化曲线。

表 5 MFCSA 在三杆桁架设计问题中的最优解

Tab. 5 Optimal solutions of MFCSA in three bar truss design

参数	参数值
x_1	0.788675136446368
x_2	0.408248285226876
g_1	-6.661338147750939E-16
g_2	-1.464101621091225
g_3	-0.535898378908775
$f(x)$	263.8958433764684

表 6 不同算法三杆桁架问题对比

Tab. 6 Comparison of different algorithms for three bar truss problems

算法	x_1	x_2	最优值
MFCSA	0.788675136446368	0.408248285226876	263.8958433764684
CSA[10]	0.7886751284	0.4082483080	263.8958433765
MVO[21]	0.78860276	0.408453070000000	263.8958499
GOA[22]	0.788897555578973	0.407619570115153	263.895881496069
MFO[23]	0.788244771	0.409466905784741	263.8959797
WCA[24]	0.788651	0.408316	263.895843
MBA[25]	0.7885650	0.4085597	263.8958522

基于表 6 实验结果,相较于其他六种算法,MFCSA 具有最好的优化精度。从图 8 上看,MFCSA 用不到 50 代就已经找到了该问题的最优解。因此,MFCSA 的收敛速度是比较快的。

4.2 压力容器设计问题

压力容器设计是为了使压力容器材料、成形、焊接成本最小。如图 9 所示,该优化问题包括四个决策变量:容器壁的厚度(x_1 或 T_s)、半球头部的厚度(x_2 或 T_h)、内半径(x_3 或 R)和圆柱截面的长度(x_4 或 L)。数学模型如下:

$$\min \quad f(x)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

$$\text{s.t.} \quad g_1(x)=-x_1+0.0193x_3\leqslant 0$$

$$g_2(x)=-x_2+0.00954x_3\leqslant 0$$

$$g_3(x)=-\pi x_3^2x_4-4\pi x_3^3/3+1296000\leqslant 0$$

$$g_4(x)=x_4-240\leqslant 0$$

$$0\leqslant x_i\leqslant 100,i=1,2;10\leqslant x_i\leqslant 200,i=3,4 \qquad (8)$$

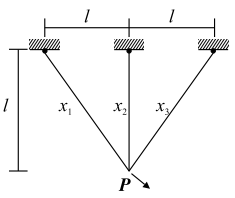

图 7 三杆桁架示意图

Fig. 7 Schematic diagram of three bar truss

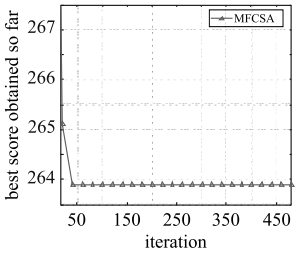

图 8 MFCSA 求解三杆桁架问题收敛曲线

Fig. 8 Convergence curve of MFCSA for solving three bar truss problem

表 7 为 MFCSA 求解该问题的结果,表 8 则是 MFCSA、CPSO[26]、HPSO[27]、PSO[2]、CSA[10]、GA3[28]、ABC[29]、CGWO[30]、STA[31] 求解决该问题的对比结果。其中,每种对比算法均独立运行 30 次,并记录最优值、最差值、平均值、标准差实验数据。表 8 中的 N/A 表示数据空缺。

表 7 MFCSA 压力容器设计最佳解

Tab. 7 Optimal solution of MFCSA in pressure vessel design

参数	参数值	参数	参数值
x_1	0.778212849328	g_1	-2.159912367e-05
x_2	0.385558851230	g_2	-8.985130461e-04
x_3	40.320790166029	g_3	-83.286298930
x_4	200.0	g_4	-40
$f(x)$	5.8884751298e+03		

表 8 不同算法压力容器设计对比

Tab. 8 Comparison of different algorithms for pressure vessel design

算法	最优值	最差值	平均值	标准差
CPSO[26]	6061.0777	6363.8041	6147.1332	86.45
HPSO[27]	6059.7143	6288.6770	6099.9323	86.20
PSO[2]	6693.7212	14076.3240	8756.6803	1492.567
CSA[10]	6059.7143	7332.8416	6342.4991	384.9454
GA3[28]	6288.7445	6 308.4970	6 293.8432	7.4133
ABC[29]	6059.7147	N/A	6245.3081	205
CGWO[30]	6134.18	6388.1109	6159.32	254.5
STA[31]	6287.0372	6377.8783	6332.2545	150.8838
MFCSA	5888.4751	6343.1780	5975.4768	186.1512

从表 8 可以看出,MFCSA 找到的最优值和平均值均好于其他对比算法;其标准差好于 CSA,但没有 CPSO、HPSO、GA3、STA 的好。图 10 是 MFCSA 求解压力容器设计问题的收敛曲线。从收敛曲线上看,MFCSA 找到该设计问题最优解的速度还是比较快的,说明可用来解决工程约束等实际问题。

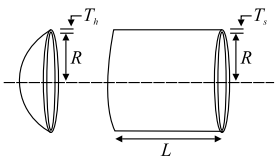

图 9 压力容器设计问题示意图

Fig. 9 Schematic diagram of pressure vessel design problems

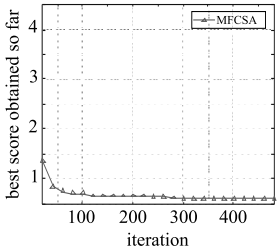

图 10 MFCSA 求解压力容器问题收敛曲线

Fig. 10 Convergence curve of MFCSA for solving pressure vessel problem

5　结束语

针对 CSA 的不足,本文提出采用多模式飞行的乌鸦搜索算法(MFCSA)。MFCSA 基于觅食能力的强弱将群体分成觅食能力较强和较弱两个组,觅食能力较强者采用尾随跟踪当前群体最优者策略,在群体信息的引导下飞到群体当前最优位置附近开展局部搜索活动;觅食能力较弱者采用观察和学习强者的觅食方法、遇到危险时则采用迅速飞离两种策略。采用观察和学习强者的觅食方法可提升算法的全局探索能力,采用迅速飞离策略的乌鸦个体根据其适应度值选择不同的搜索区域,保持了种群的多样性,增强了算法的探索能力。通过 15 个基准测试函数和 2 个工程应用问题的实验测试与仿真结果表明,本文算法在优化精度、收敛速度方面得到了较大的提升,规避陷入局部最优的能力得到明显增强,算法稳定性更好。

参考文献:

[1] Holland, John H. Genetic algorithms and the optimal allocation of trials[J]. SIAM Journal on Computing,1973,2(2):88-105.

[2] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proc of International Conference on Neural Networks. Piscataway, NJ: IEEE Press,1995:1942-1948.

[3] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents[J]. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics,1996,26(1):29-41.

[4] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization,2007,39(3):459-471.

[5] 李晓磊,邵之江,钱积新. 一种基于动物自治体的寻优模式:鱼群算法[J]. 系统工程理论与实践,2002,22(11):32-38. (Li Xiaolei, Shao Zhijiang, Qian Jixin. An optimizing method based on autonomous animats: fish-swarm algorithm [J]. System Engineering-Theory & Practice,2002,22(11):32-38.)

[6] Yang Xinshe. A new metaheuristic bat-inspired algorithm[M]//Nature Inspired Cooperative Strategies for Optimization. Berlin: Springer, 2010:65-74.

[7] Rui Tang, Simon F, Yang Xinshe, *et al*. Wolf search algorithm with ephemeral memory[C]//Proc of the 7th International Conference on Digital Information Management. Piscataway, NJ: IEEE Press, 2012: 165-172.

[8] Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems[J]. Neural Computing and Applications,2016,27(4): 1053-1073.

[9] Javidy B, Hatamlou A, Mirjalili S. Ions motion algorithm for solving optimization problems[J]. Applied Soft Computing,2015,32(7): 72-79.

[10] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm[J]. Computers & Structures,2016,169(6):1-12.

[11] Balochian S, Baloochian H. Social mimic optimization algorithm and engineering applications [J]. Expert Systems with Application, 2019,134(11):178-191.

[12] Rizk-Allah R M, Hassanien A E, Bhattacharyya S. Chaotic crow search algorithm for fractional optimization problems [J]. Applied Soft Computing,2018,10(71):1161-1175.

[13] Farid M, Hamdi A. A modified crow search algorithm (MCSA) for solving economic load dispatch problem[J]. Applied Soft Computing,2018,71(10):51-65.

[14] 葛知著,张达敏,张琳娜,等. 混合策略改进的乌鸦搜索算法[J]. 计算机应用研究,2021,38(11):3334-3339. (Ge Zhizhu, Zhang Damin, Zhang Linna, *et al*. Crow search algorithm improved by mixed strategy[J]. Application Research of Computers,2021,38(11):3334-3339.)

[15] 肖子雅,刘升,韩斐斐,等. 正弦余弦指引的乌鸦搜索算法研究[J]. 计算机工程与应用,2019,55(21):51-58. (Xiao Ziya, Liu Sheng, Han Feifei, *et al*. Crow search algorithm based on directing of sine cosine algorithm[J]. Computer Engineering and Applications,2019,55(21):51-58.)

[16] Qu Chiwen, Fu Yanming. Crow search algorithm based on neighborhood search of non-inferior solution set[J]. IEEE Access,2019,7: 52871-52895.

[17] 霍林,郭雅蓉,覃志健. 具有自适应步长的柯西变异乌鸦算法[J]. 计算机科学,2020,47(12):218-225. (Huo Lin, Guo Yarong, Qin Zhijian. Crow search algorithm with cauchy mutation and adaptive step size[J]. Computer Science,2020,47(12):218-225.)

[18] 辛梓芸,张达敏,陈忠云,等. 多段扰动的共享型乌鸦算法[J]. 计算机工程与应用,2020,56(2):55-61. (Xin Ziyun, Zhang Damin, Chen Zhongyun, *et al*. Shared crow algorithm using multi-segment perturbation[J]. Computer Engineering and Applications,2020, 56(2):55-61.)

[19] 赵世杰,高雷阜,于冬梅,等. 基于变因子加权学习与邻代维度交叉策略的改进 CSA 算法[J]. 电子学报,2019,47(1):40-48. (Zhao Shijie, Gao Leifu, Yu Dongmei, *et al*. Improved crow search algorithm based on variable-factor weighted learning and adjacent-generations dimension crossover strategy[J]. Acta Electronica Sinica,2019,47(1):40-48.)

[20] 林忠甫,颜力,黄伟,等. 基于参数自适应策略的改进乌鸦搜索算法[J]. 计算机科学,2021,48(S1):260-263,284. (Lin Zhongfu, Yan Li, Huang Wei, *et al*. Improved crow search algorithm based on parameter adaptive strategy[J]. Computer Science,2021,48(S1): 260-263,284.)

[21] Mirjalili S, Mirjalili S M, Hatamlou A. Multi-verse optimizer: a nature-inspired algorithm for global optimization[J]. Neural Computing and Applications,2015,27(2):495-513.

[22] Saremi S, Mirjalili S, Lewis A. Grasshopper optimisation algorithm: theory and application [J]. Advances in Engineering Software, 2017,105(3):30-47.

[23] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm[J]. Knowledge-Based Systems,2015, 89(11):228-249.

[24] Zheng Yujun. Water wave optimization: a new nature-inspired metaheuristic[J]. Computers & Operations Research,2015,55(3): 1-11.

[25] Sadollah A, Bahreininejad A, Eskandar H, *et al*. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems[J]. Applied Soft Computing Journal,2013,13(5):2592-2612.

[26] He Qie, Wang Ling. An effective co-evolutionary particle swarm optimization for constrained engineering design problem[J]. Engineering Applications of Artificial Intelligence,2007,20(1):89-99.

[27] He Qie, Wang Ling. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization[J]. Applied Mathematics and Computation,2007,186(2):1407-1422.

[28] Coello C A C. Use of a self-adaptive penalty approach for engineering optimization problems[J]. Computers in Industry,2000,41(2): 113-127.

[29] Akay B, Karaboga D. Artificial bee colony algorithm for large-scale problems and engineering design optimization[J]. Journal of Intelligent Manufacturing,2012,23(4):1001-1014.

[30] 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.

[31] Han Jie, Yang Chunhua, Zhou Xiaojun, *et al*. A two-stage state transition algorithm for constrained engineering optimization problems[J]. International Journal of Control, Automation and Systems, 2018,16(2):522-534.