DOI:10.19322/j.cnki.issn.1006-4710.2022.01.010

基于多目标的竞争粒子群优化算法的研究

王 彬[1]，王丹妮[1]，江巧永[1]，杨家杰[2]

(1.西安理工大学 计算机科学与工程学院，陕西 西安 710048；
2.滑铁卢大学 数学院，加拿大 滑铁卢 N2L 3G1)

摘要：针对竞争粒子群优化算法较易陷入局部最优及对全局搜索能力有限等问题，本文提出基于多目标的竞争粒子群优化算法(MOCSO)。首先，为提高算法的多样性，建立多目标模型，在二元锦标竞争机制下分别以适应度值与多样性作为粒子优劣的评判目标，将粒子划分到不同的非支配层中；其次，为有效兼顾粒子的探索与开发，引入概率随机选择策略，有效保留种群内的优秀基因，引导种群的进化方向。实验结果表明，在CEC2008基准测试函数集上与其他6种算法进行性能与计算效率的比较，本算法具有一定的优势。

关键词：竞争粒子群优化算法；多目标模型；概率随机选择；适应度值；多样性

中图分类号：O224 **文献标志码**：A **文章编号**：1006-4710(2022)01-0075-08

Research on competitive particle swarm optimization algorithm based on multi-objective

WANG Bin[1], WANG Danni[1], JIANG Qiaoyong[1], YANG Jiajie[2]

(1. Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China;
2. Faculty of Mathematics, University of Waterloo, Waterloo N2L 3G1, Canada)

Abstract: For the problem that competitive swarm optimization is easy to fall into local optimization and limited global search ability, this paper proposes a multi-objective competitive particle swarm optimization (MOCSO) algorithm. Firstly, in order to improve the diversity of the algorithm, a multi-objective model is established. By the competition mechanism of binary championship, the fitness value and diversity are taken as the evaluation objectives of the particle, and the particles are divided into different non-dominated layers. Secondly, in order to effectively consider the exploration and exploitation of particles, a probabilistic random selection strategy is introduced, effectively retaining the excellent genes in the population and guides the evolutionary direction of the population. The experimental results show that this algorithm has some advantages compared with six other algorithms in performance and computational efficiency on CEC2008 benchmark function.

Key words: competitive particle swarm optimize; multi-objective model; probabilistic random selection; fitness value; diversity

近年来，随着时代的发展与科技的进步，出现的许多复杂问题都已无法通过简单的模型求解，这些问题的决策变量多、问题规模大、数学性质复杂，被归类称为大规模问题[1]。因此，设计高效可靠的算法解决大规模问题是十分必要的。

针对这些优化问题，学者们提出了大量的启发式随机优化算法，进化算法就是在这种背景下提出的。受自然界生物进化的启发，如帝王蝶优化算法(MOB)[2]、大象放牧优化算法(EHO)[3]、磷虾群算法(KH)[4]等，因其操作简单、优化性能强的特点，在过去几十年中得到了快速发展，但是，进化算法在优化大规模问题时仍存在一些不足，因此，很有必要利用算法的学习能力[5]，研究出高效可靠的算法来解决大规模优化问题。

收稿日期：2021-04-12； **网络出版日期**：2021-07-14

网络出版地址：https://kns.cnki.net/kcms/detail/61.1294.n.20210714.1006.002.html

基金项目：国家自然科学基金资助项目(61976177)

通信作者：王彬，男，硕士，副教授，研究方向为进化计算、人工智能理论与应用。E-mail:wb@xaut.edu.cn

目前,针对大规模问题的进化算法主流解决策略[6]分为三类:一是静态的协同进化,即将复杂的高维问题分解成简单的低维问题进行优化求解;二是动态的协同进化,即利用分组策略将不可分的决策变量分在不同组,给予不同组一个权重向量[7],将对问题的优化转化为对权重的优化。前两种解决策略都要在算法的前期求解决策变量之间的相关性,这样会耗费大量的时间和精力。三是整体进化,这是一种不使用分组策略的算法,主要是针对大规模问题的特点,在原有的算法中加入一些搜索策略或者是利用新的进化机制等方式来提升原有算法的寻优能力。

竞争粒子群优化算法(CSO)[8]就是整体进化的新兴代表,由于该算法结构简单、性能良好,因此在求解工程类优化问题时得到了广泛应用,且备受关注。为进一步提高算法的性能,不少科研人员都基于该算法进行了改进。2019年,Mohapatra等[9]进一步提出了ICSO,使用的是三竞争机制,它将这种遗传概念与CSO结合起来,以便更快地收敛。2020年,Zhang等[10]提出了CSO-MA,它通过变异种群中的输家来增加种群多样性,提高探索能力。同年,Xiong等[11]提出了WLCSODGM,其对优胜者领先的搜索策略和动态高斯变异算子的引入,有效地解决了传统CSO容易陷入局部最优的弊端。

然而,已有的CSO算法通常是以适应度值的大小作为参考依据来判断算子的优劣,但是,衡量粒子群算法性能的关键标准有两个:收敛速度和全局搜索能力。因此,如何平衡粒子收敛速度和全局搜索能力是有效改进算法的关键。

基于以上考虑,本文通过主动学习种群的多样性来构建多目标模型,在原有的评判基础上,将多样性也作为评估粒子优劣的条件,从而平衡种群的探索与开发能力。最后,在测试函数集CEC2008上,将本文所提算法与其他改进算法进行对比,验证本文算法的有效性。

1 相关理论

1.1 竞争粒子群优化算法

粒子群优化算法(PSO)是基于模仿群居动物的群居行为而产生的算法,其概念简单且搜索效率较高。在PSO的基础上,CSO引入了生物学中的适者生存竞争机制,不用在每一次更新时都要记住个体最优值与全局最优值。它是在一个种群内采用成对竞争机制,使用两两比较的模式更新种群。

一般情况下,考虑的都是问题最小化:

$$\min f(X)\ \text{s.t.}\ x \in X \tag{1}$$

式中:$X \in R^n$ 是可行解解集;n 表示搜索空间的维度,即决策变量的数量。

为了解决PSO所存在的弊端,首先初始化一个包含 m 个粒子的种群 $p(t)$,其中,m 是种群规模大小,t 是迭代次数。另外,每个粒子都有 n 维位置。上述问题的候选解用 $\boldsymbol{X}_i(t)=(x_{i,1}(t),x_{i,2}(t),\cdots,x_{i,n}(t))$ 表示,$\boldsymbol{V}_i(t)=(v_{i,1}(t),v_{i,2}(t),\cdots,v_{i,n}(t))$ 代表 n 维速度向量。在每一代中,粒子种群 $p(t)$ 被随机分为 $\frac{m}{2}$ 对,然后任选两个粒子根据其适应度值进行竞争,每一次比赛的结果中,适应度值较小的称为赢家,直接进入下一代 $p(t+1)$,而适应度值较大的输家则需要向赢家学习并更新自己的位置和速度,才能进入下一代,当所有的粒子都传递给下一代时,意味着一次迭代完成。

竞争后输家的更新公式为:

$$\boldsymbol{V}_{l,j}(t+1)=\boldsymbol{R}_1(k,t)\boldsymbol{V}_{l,j}(t)+\boldsymbol{R}_2(\boldsymbol{X}_{w,j}(t)-\boldsymbol{X}_{l,j}(t))+\varphi\boldsymbol{R}_3(k,t)(\overline{\boldsymbol{X}_j(t)}-\boldsymbol{X}_{l,j}(t)) \tag{2}$$

$$\boldsymbol{X}_{l,j}(t+1)=\boldsymbol{X}_{l,j}(t)+\boldsymbol{V}_{l,j}(t+1) \tag{3}$$

式中:$\boldsymbol{X}_{w,k}(t)$、$\boldsymbol{X}_{l,k}(t)$、$\boldsymbol{V}_{w,k}(t)$、$\boldsymbol{V}_{l,k}(t)$ 分别为第 t 代、第 k 次迭代的赢家和输家的位置和速度;$\boldsymbol{R}_1(k,t)$、$\boldsymbol{R}_2(k,t)$、$\boldsymbol{R}_3(k,t)$ 是3个分布在 $[0,1]$ 的随机向量;$\overline{\boldsymbol{X}_j(t)}$ 表示第 t 代粒子的平均位置;φ 是调节影响 $\overline{\boldsymbol{X}_j(t)}$ 的控制参数。

式(2)为速度更新公式,式(3)为位置更新公式;式(2)中的 $\boldsymbol{R}_1(k,t)\boldsymbol{V}_{l,j}(t)$ 是为了确保搜索过程的稳定性;$\boldsymbol{R}_2(\boldsymbol{X}_{w,j}(t)-\boldsymbol{X}_{l,j}(t))$ 表示竞争中失败者向成功者学习的过程,异于经典的粒子群进化算法(PSO),这种机制不需要所有粒子记住它们过去经历的最优位置;$\varphi\boldsymbol{R}_3(k,t)(\overline{\boldsymbol{X}_j(t)}-\boldsymbol{X}_{l,j}(t))$ 是要求失败的粒子向当前种群的平均位置学习,反映了粒子间的协同合作。

1.2 多目标优化

实际应用中,通常会遇到多个目标需要同时优化的问题,这类问题统称为多目标优化问题[12](multiobjective optimization problem, MOP),其数学公式为:

$$\begin{cases}\min \boldsymbol{F}(x)=(f_1(x),f_2(x),\cdots,f_m(x))\\ \text{s.t.}\ g_j(x)\leqslant 0,j=1,2,\cdots,q\\ h_k(x)=0,k=1,2,\cdots,p\end{cases} \tag{4}$$

式中:$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 被称为决策变量,也是 n 维的决策空间;$f_i(x)(i=1,2,\cdots,m)$ 是第 i 个被最小化的目标;$g_j(x)(j=1,2,\cdots,q)$ 定义为第 j 个不等式约束;$h_k(x)(k=1,2,\cdots,p)$ 定义为第 k 个等

式约束。此外，所有的约束决定了可行解的集合 Ω，$Y=\{F(x)\mid x\in\Omega\}$ 被称为目标空间。

对于多目标问题而言，一个问题的优化可能会导致另一个问题的退化，因此，对于多目标问题的优化，很难找到一个同时满足各个问题最优值的解，而是找到一个平衡各个目标的 Pareto 最优解集。

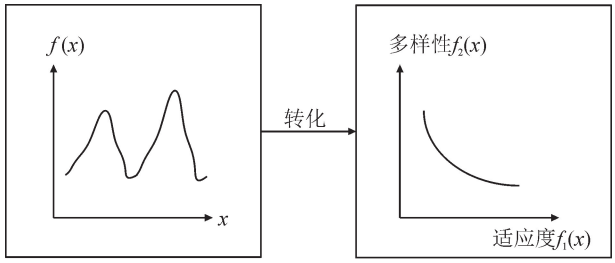
2 研究动机

2.1 CSO 在多样性方面存在的问题

收敛性分析是刻画竞争粒子群优化算法动力学行为的一个重要方面，同时也是确保竞争粒子群优化算法可靠性的基本前提。文献[8]证明发现，CSO 具有良好的收敛性能，但却无法保证种群收敛到全局最优，且由于 CSO 的二元锦标竞争机制，每次迭代只依赖一半的粒子在搜索空间中进行开发，对种群的搜索能力有很大的限制。因此，有必要通过提高种群多样性来确保种群尽可能地收敛到全局最优。

2.2 解决思路

为了有效平衡种群的探索与勘探能力，在原有 CSO 以适应度值作为评价标准的基础上，加入另一个评价标准——多样性，这样不仅能兼顾个体的收敛速度，同时也能考虑到全局搜索能力。如图 1 所示，多样性的引入，将原本的单目标模型转化为多目标模型。



(a) 单目标模型 (b) 多目标模型

图 1 单目标模型转化为多目标模型

Fig. 1 Single objective model transformed into a multi-objective model

3 基于多目标的竞争粒子群优化算法

基于以上考虑，本节提出了一种基于多目标的竞争粒子群优化算法（MOCSO），不同于已有的 CSO，MOCSO 引入了多目标模型，将原本的单目标模型转化为多目标模型，从而在保证原有收敛性的基础上，提高了算法的开发能力，有效平衡了种群的开发与探索能力。此外，选择策略增加了随机性，在引导种群进化方向的同时，可防止种群陷入局部最优。

3.1 构建多目标模型

事实上，种群中个体的价值不仅仅取决于它的适应度值，当它与其它个体交配时，个体在解空间中产生足够分散的后代的能力也是非常重要的，因为它有助于有效且详尽地探索解空间。因此，在种群进化过程中，要尽可能使个体的探索能力和开发能力达到一个平衡，既要考虑尽快地达到全局最优解，也要考虑个体对种群多样性的影响。基于以上原因，本文采用多目标模型。模型如图 2 所示，将种群中的所有个体按照支配关系划分在不同的非支配层中，很明显，第一层的个体要比第二层的个体表现性能好，以此类推。

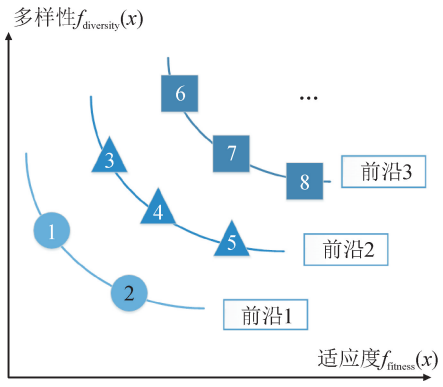


图 2 多目标模型的非支配排序

Fig. 2 Non dominated sorting of multi-objective

以个体适应度值函数与个体多样性度量函数为横纵轴，表示要优化的两个目标函数：

$$\text{适应度：}f_{\text{fitness}}(x_i)=f(x_i) \tag{5}$$

$$\text{多样性：}f_{\text{diversity}}(x_i)=\min_{i\neq j}\|x_i-x_j\| \tag{6}$$

由于个体适应度值越小，代表其越接近种群最优解，而多样性值越大，有利于在种群更新时生成尽可能分散的解，防止算法陷入局部最优。因此，与种群中与每个个体有关的两项目标可界定为：

$$\min f_{\text{fitness}}(x_i) \tag{7}$$

$$\min(-f_{\text{diversity}}(x_i)) \tag{8}$$

其中，多样性采用欧氏距离，因其操作简单，易于计算。

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{9}$$

采用非支配排序，将种群中的所有个体都划分在不同的非支配层中，每一层的个体性能整体都要优于后一层的，但在同一层面的个体都相互为非支配关系。

3.2 概率随机选择策略

非支配排序已经将个体进行优劣划分，为了有效平衡种群的探索与开发能力，在 CSO 中引入基于

排序的选择策略[13],通过概率确定当前的输家和赢家。目的是让表现良好的个体仍能保留自己的优秀基因,从而有效地控制种群的进化方向,加快种群收敛速度。但表现较劣的个体也有可能被选为赢家,这样有利于增加种群多样性,防止陷入局部最优。详细策略步骤如下。

1) 编号分配

环境选择策略考虑了个体对种群收敛性和多样性贡献的大小,并按照其在 Pareto 前沿上的顺序进行全排序。其中,性能表现良好的个体排在首位,性能较差的个体排在末位,即越优质的个体,指标就越小。为了以较大的概率选择性能较好的个体作为赢家,需要重新定义种群编号。可表示为:

$$R_i = NP + 1 - i, i = 1, 2, \cdots, NP \tag{10}$$

式中:NP 是种群规模;i 是当前种群中第 i 个个体的指标。

2) 选择概率

根据式(10)中对个体编号的计算,第 i 个个体被选择为赢家的选择概率为:

$$P_i = \frac{R_i}{NP}, i = 1, 2, \cdots, NP \tag{11}$$

因此,适应度值较低和多样性较好的个体将有很大的概率在排序种群中被选择作为赢家。

3.3 算法描述

基于以上对 MOCSO 的描述,多目标的竞争粒子群优化算法(MOCSO)的算法流程如表 1 所示。与经典 CSO 相比,MOCSO 主要增加了一个双目标模型(step2.1 和 step2.2)和选择策略(step2.4)。

4 实验及分析

4.1 对比算法

为验证所提算法的有效性,本节引入 6 种对比算法,分别是:竞争粒子群优化算法(CSO)[8]、继承的竞争粒子群优化算法(ICSO)[14]、改良的竞争粒子群优化算法(MCSO)[15]、动态分组的竞争粒子群优化算法(CSO-DG)[16]、反向学习的竞争粒子群优化算法(OBLCPSO)[17]、排序学习的竞争离子群优化算法(RBLSO)[18]。

4.2 测试函数

在 CEC2008 上对 MOCSO 进行数值实验,为了确保运算的准确性,每个算法在每个测试函数上独立运行 51 次。CEC2008 的测试函数主要分为两类:单模测试函数($F_1 \sim F_2$)、多模测试函数($F_3 \sim F_7$)。

表 1 MOCSO 的算法流程

Tab.1 Flow chart of MOCSO

算法 1:基于多目标的竞争粒子群优化算法(MOCSO)

输入:N:种群规模;
D:决策变量的维数;
Gen:更新代数;
L_j:第 j 维决策变量的上限;
U_j:第 j 维决策变量的下限;
$MaxFES$:最大函数评价次数;

输出:迭代次数和最优值

Step 1)**初始化**

Step 1.1)**初始化种群**:在种群中随机生成 N 个个体,每个个体携带 D 维信息;

Step 1.2)**初始化最优值**:求解初始化种群中每个个体的适应度值,并将最小适应度值赋值给 *bestever*;

Step 1.3)**初始化速度**:设置一个 $N \times D$ 维的零矩阵存放初始化种群的初始速度;

Step 1.4)**初始化函数评价次数**:设置 $FES = N$;

Step 2)**演化**

Step 2.1)**计算欧式距离**:利用欧式距离求解每个个体与其他个体之间的拥挤距离;

Step 2.2)**构建双目标模型**:将适应度值,拥挤距离作为两个目标;

Step 2.3)**非支配排序**:对种群个体进行非支配排序,得到一组 Pareto 解集;

Step 2.4)**环境选择**:利用选择策略确定输赢家,赢家直接进入下一代,输家则向赢家学习并进入下一代;

Step 2.5)**移除个体**:在当代中移除这两个个体;

Step 3)**停止准则**

若满足终止条件,则停止整个循环,并输出迭代次数和最优值;否则,转到 Step 2)。

4.3 参数设置

所有仿真实验的测试平台都是相同的,均为 2.60 GHz CPU 和 4 GB RAM,Microsoft Windows 10 操作系统,Intel(R) Core™ i5-3230M;测试软件是 Microsoft Windows 10 操作系统下的 MATLAB 2016a。

为评估 MOCSO 的有效性,将 MOCSO 与现有 CSO 的不同改进算法进行比较,其公共参数设置如表 2 所示。

表 2 对比算法中的公共参数设置

Tab. 2 Public parameter setting in comparative algorithm

参数设置
每个测试函数的维度：$D=30/50/100$
种群规模：$N=100$
最大函数评价次数（$MaxFES$）：$MaxFES=D\times 10\,000$
每个测试函数独立运行的次数（$runmax$）：$runmax=51$

此外，本文采用两个标准来评估算法的性能：

1）收敛速度：绘制了不同类型测试函数的收敛曲线，直观地显示出算法的收敛情况，并反映出最优解在各个算法之间的误差值；

2）统计分析：为了更客观地比较两种算法的性能，本文采用 0.05 显著性水平的 Wilcoxon 秩和检验对实验结果进行统计和分析，如果得到的概率值小于 0.05，则认为两种算法的性能有显著差异，用 1 表示，否则为无显著差异，用 0 表示。

4.4 实验结果及分析

表 3、表 4 和表 5 分别记录了 MOCSO 与 6 组对比算法在 CEC2008 测试函数集上的相关实验结果，并给出了统计分析性能比较。从最后一行的“$w/s/l$”值可以看出，MOCSO 与其他算法存在显著差异，其在大多数基准测试函数上的整体效果明显优于其他算法。

进一步分析可知，MOCSO 在每个测试函数中的排名一般位于前两名，整体收敛效果良好，这是因为在评判标准中加入了多样性，又保留了原始算法中粒子优秀的探索能力，同时考虑到粒子的勘探能力，使得种群在更新过程中能有效平衡粒子收敛速度和全局搜索能力。

表 3 MOCSO 与 6 组对比算法在 30 维的 CEC2008 上独立运行 51 次的统计结果

Tab. 3 Statistical results of MOCSO and 6 groups of compared algorithms running 51 times independently on 30 dimensional CEC2008

函数	MOCSO	CSO	差异	ICSO	差异	CSO-DG	差异	MCSO	差异	OBLCPSO	差异	RBLSO	差异
F_1	5.02E-27 (0)	1.63E-27 (0)	0	0.051 76 (0)	0	**1.54E-27** (0)	0	1.91E-27 (0)	0	13 352.82 (5.43E+03)	1	0 (0)	0
F_2	**85.04** (8.61E-14)	85.04 (8.61E-14)	0	85.04 (8.61E-14)	0	85.04 (8.61E-14)	0	85.04 (8.61E-14)	0	85.04 (1.99E-04)	1	85.04 (8.61E-14)	0
F_3	88.04 (148.47)	**50.57** (75.78)	1	339.22 (38.10)	1	98.88 (230.44)	1	129.117 4 (99.29)	1	1.28E+09 (6.95E+08)	1	120.575 (120.53)	1
F_4	**11.18** (5.28)	14.01 (1.85)	1	152.79 (2.15)	1	13.91 (3.15)	0	15.96 (2.28)	1	308.958 5 (23.00)	1	17.948 3 (2.64)	1
F_5	0.000 193 (0)	0.000 15 (0)	0	0.161 1 (0)	0	0.000 15 (0)	0	0.000 483 (0)	0	107.13 (37.11)	1	**0** (0)	0
F_6	1.05E-14 (1.06E-15)	1.38E-14 (1.75E-15)	1	0.100 5 (0)	1	1.34E-14 (1.16E-15)	1	1.42E-14 (1.66E-15)	1	19.43 (1.25)	1	**3.41E-15** (1.37E-15)	0
F_7	−365.05 (47.64)	**−460.85** (5.88)	1	−323.82 (3.93)	1	−457.80 (7.52)	1	−455.90 (5.78)	1	−319.78 (13.25)	1	−454.90 (57.15)	1
$w/s/l$		2/3/2		4/3/0		2/4/1		3/3/1		7/0/0		2/4/1	

注：1 表示有显著差异，0 表示无显著差异；括号内外数值分别表示多次测试结果的方差、均值；粗体数值表示每个测试函数排名靠前的结果；“w”表示该算法优于其他算法，“s”表示两种算法的性能相似，“l”表示该算法的性能逊于其他算法；下同。

表 4 MOCSO 与 6 组对比算法在 50 维的 CEC2008 上独立运行 51 次的统计结果

Tab. 4 Statistical results of MOCSO and 6 groups of compared algorithms running 51 times independently on 50 dimensional CEC2008

函数	MOCSO	CSO	差异	ICSO	差异	CSO-DG	差异	MCSO	差异	OBLCPSO	差异	RBLSO	差异
F_1	**0** (0)	1.08E- 26 (1.73E-27)	1	7.20E-27 (9.29E-28)	1	1.09E-26 (0)	0	1.33E-26 (1.84E-14)	1	3.44E+04 (9.14E+03)	1	0 (0)	0
F_2	**86.77** (4.31E-14)	86.77 (4.30E-14)	0	86.77 (4.31E-14)	0	86.77 (4.31E-15)	0	86.77 (4.31E-14)	0	86.95 (0.18)	1	86.77 (4.31E-14)	0
F_3	47.41 (13.27)	42.87 (28.95)	1	**40.16** (27.32)	1	43.02 (34.88)	1	45.94 (46.12)	1	5.12E+09 (2.69E+09)	1	45.25 (21.41)	1
F_4	**25.65** (4.67)	29.17 (4.76)	1	26.77 (5.07)	0	31.94 (5.30)	1	36.71 (7.38)	1	621.65 (35.88)	1	36.31 (6.49)	1
F_5	**0** (0)	0.01 (0)	0	2.90E-04 (0.01)	0	2.90E-04 (0)	0	0.01 (0.01)	0	300.84 (90.94)	1	3.87E-04 (0.01)	0
F_6	**3.83E-15** (9.65E-16)	2.19E-14 (2.96E-15)	1	1.72E-14 (2.50E-15)	1	2.22E-14 (1.79E-15)	1	2.68E-14 (3.11E-15)	1	20.00 (0.812 2)	1	6.33E-15 (1.48E-15)	1
F_7	−636.74 (123.46)	−788.52 (9.57)	1	−789.46 (9.88)	1	**−789.80** (8.76)	1	−782.28 (10.33)	1	−536.39 (21.05)	1	−787.59 (11.76)	1
$w/s/l$		3/2/2		2/3/2		2/3/2		3/2/2		7/0/0		2/3/2	

表 5 MOCSO 与 6 组对比算法在 100 维的 CEC2008 上独立运行 51 次的统计结果

Tab. 5 Statistical results of MOCSO and 6 groups of compared algorithms running 51 times independently on 100 dimensional CEC2008

函数	MOCSO	CSO	差异	ICSO	差异	CSO-DG	差异	MCSO	差异	OBLCPSO	差异	RBLSO	差异
F_1	**0** (0)	0 (0)	0	5.56E-26 (5.16E-27)	1	1.71E-29 (5.01E-29)	1	1.14E-25 (1.24E-26)	1	1.1816E+05 (1.54E+04)	1	0 (0)	0
F_2	**89.65** (0)	89.65 (0)	0	89.65 (0)	0	89.65 (0)	0	89.65 (0)	0	86.68 (0.027 6)	1	89.65 (0)	0
F_3	796.28 (677.45)	140.60 (151.57)	1	**131.06** (111.46)	1	135.45 (100.89)	1	230.20 (293.81)	1	1.76E+10 (4.67E+09)	1	252.08 (338.87)	1
F_4	80.47 (14.16)	**26.38** (4.89)	1	70.12 (8.79)	0	56.37 (7.17)	1	102.87 (16.59)	1	1.36E+03 (45.18)	1	101.13 (17.16)	1
F_5	**0** (0)	0 (0)	0	7.73E-04 (0.01)	1	5.80E-04 (0.01)	0	7.73E-04 (0.01)	0	944.46 (132.11)	1	0 (0)	0
F_6	**7.11E-15** (0.27)	1.03E-14 (1.07E-15)	1	3.38E-14 (2.96E-15)	1	1.02E-14 (1.23E-15)	1	4.90E-14 (3.28E-15)	1	20.396 5 (0.11)	1	1.06E-14 (2.80E-15)	1
F_7	−1 333.58 (240.35)	**−1 496.89** (9.36)	1	−1 466.06 (13.94)	1	−1 472.35 (18.43)	1	−1 450.95 (18.34)	1	−937.07 (27.55)	1	−1 459.36 (18.29)	1
$w/s/l$		1/3/3		3/1/3		3/1/3		4/1/2		7/0/0		2/3/2	

从秩和检验的结果来看，MOCSO 在 30 维、50 维和 100 维上，有超过一半的基准函数其性能明显优于其他对比算法，为了更直观地比较 MOCSO 与 6 组对比算法的收敛效果，图 3 给出了 MOCSO 在

测试函数集 CEC2008 上与其他算法的收敛速度的对比图，从图 3 中也可以看出，MOCSO 的整体收敛效果还是具有一定优势的。

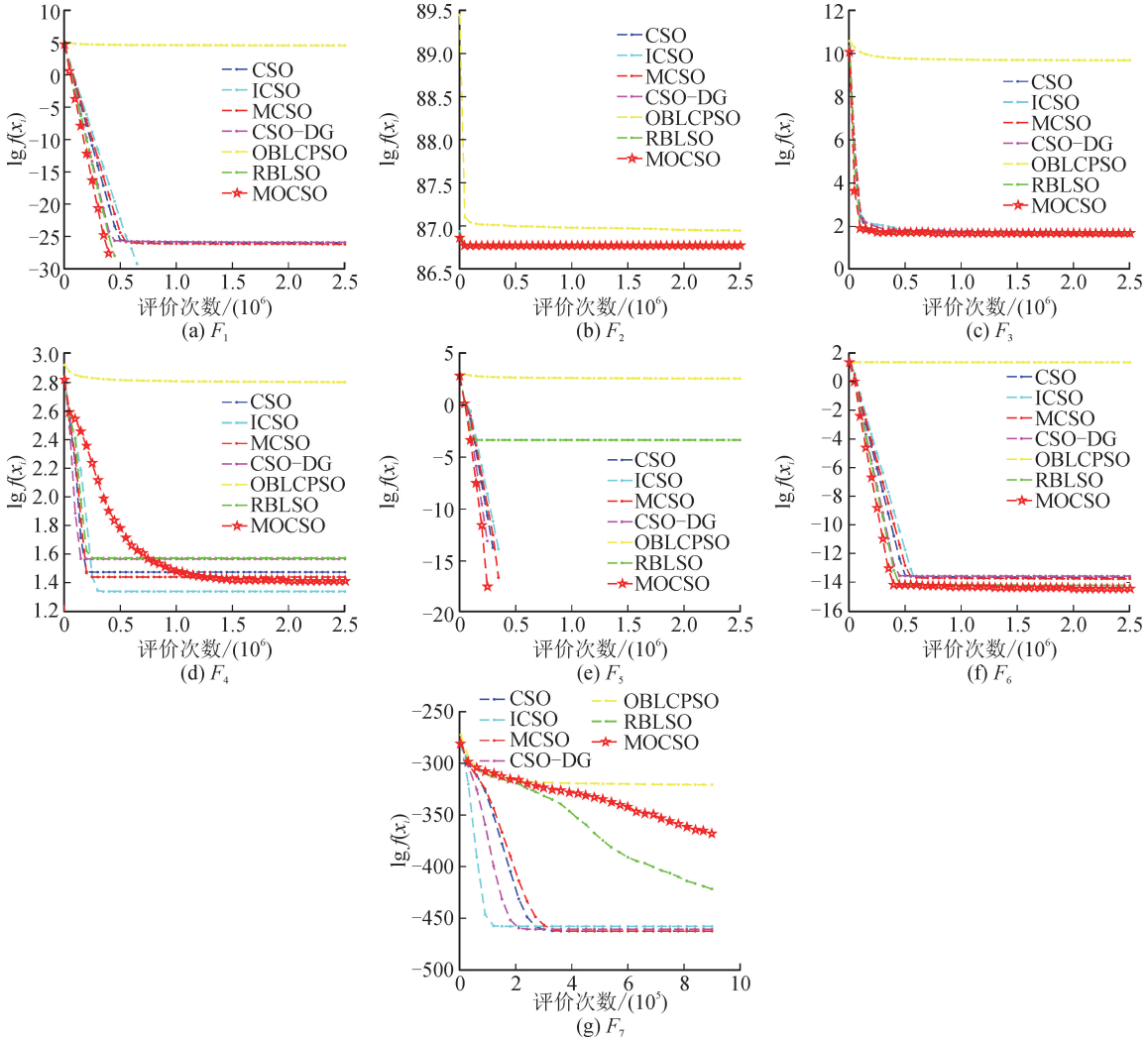


图 3　MOCSO 与 6 组对比算法在 50 维的 7 个测试函数上的收敛曲线

Fig. 3　Convergence curves of MOCSO and 6 groups of compared algorithms for 7 test functions of 50 dimensions

4.5　计算效率分析

通过比较 MOCSO 与其他 6 组算法的复杂度来分析 MOCSO 的计算效率，结果如表 6 所示。

表 6　7 组算法的复杂度对比

Tab. 6　Complexity comparison of seven groups of algorithms

算法	复杂度
CSO	$O(mn)$
ICSO	$O(mn)$
MCSO	$O(mn)$
CSO-DG	$O(mn)$
OBLCPSO	$O(mn)$
RBLSO	$O(mn)$
MOCSO	$O(m^2n)$

注：m 表示种群大小；n 表示搜索空间维度。

由表 6 可知，MOCSO 的计算成本要高于其他改进算法，因为 MOCSO 设置了多目标，因此每次迭代需要消耗大量额外的计算资源，为降低成本，该算法还有待进一步改进。

5　结　论

为了提高竞争粒子群优化算法（CSO）的性能，进一步平衡种群中粒子的探索能力与开发能力，本文提出了一种基于多目标的竞争粒子群优化算法，它在原有 CSO 的基础上提出了以下改进：

1）设计多目标演化机制，在演化过程中考虑了适应度值和拥挤距离，即在兼顾原有算法收敛速度的同时，又保留了种群的全局搜索能力；

2）引入随机选择策略，在每一次迭代中选择输

赢家,这种策略在很大程度上保存了种群中的优秀基因。

在CEC2008测试函数集上对MOCSO算法进行数值实验,并与CSO、ICSO、CSO-DG、MCSO、OBLCPSO、RBLSO进行性能对比。实验结果表明,与现有CSO及其改进方法相比,MOCSO在测试函数集上的整体效果明显占优。

通过仿真实验,虽然验证了MOCSO具有一定的有效性,但由于该算法设置了双目标,使得计算效率下降,其后续还有待进一步改进。

参考文献:

[1] 关世伟. 大规模全局优化问题的高效算法研究[D]. 西安:西安电子科技大学,2018.
GUAN Shiwei. Efficient algorithms for large scale global optimization[D]. Xi'an: Xidian University, 2018.

[2] FENG Yanhong, DEB S, WANG G G, et al. Monarch butterfly optimization: a comprehensive review[J]. Expert Systems with Applications, 2021, 168: 114418.

[3] LI Juan, LEI Hong, ALAVI A H, et al. Elephant herding optimization: variants, hybrids, and applications [J]. Mathematics,2020,8, 1415.

[4] WANG G G, GANDOMI A H, ALAVI A H, et al. A comprehensive review of krill herd algorithm: variants, hybrids and applications[J]. Artificial Intelligence Review, 2019, 51:119-148.

[5] LI Wei, WANG Gaige, GANDOMI A H. A survey of learning-based intelligent optimization algorithms [J]. Archives of Computational Methods in Engineering, 2021,28(5):3781-3799.

[6] 梁静,刘睿,于坤杰,等. 求解大规模问题协同进化动态粒子群优化算法[J]. 软件学报, 2018, 29(9): 2595-2605.
LIANG Jing, LIU Rui, YU Kunjie, et al. Dynamic multi-swarm particle swarm optimization with cooperative coevolution for large scale global optimization[J]. Journal of Software, 2018, 29(9): 2595-2605.

[7] DAS I, DENNIS J E. Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems[J]. SIAM Journal on Optimization, 1998, 8(3): 631-657.

[8] CHENG Ran, JIN Yaochu. A competitive swarm optimizer for large scale optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(2):191-204.

[9] MOHAPATRA P, DAS K N, ROY S. Inherited competitive swarm optimizer for large-scale optimization problems: theory and applications, ICHSA 2018 [M]. Singapore:Springer, 2019.

[10] ZHANG Zizhao, WONG W K, TAN K C. Competitive swarm optimizer with mutated agents for finding optimal designs for nonlinear regression models with multiple interacting factors[J]. Memetic Computing, 2020, 12(3):219-233.

[11] XIONG Guojiang, ZHANG Jing, SHI Dongyuan, et al. Winner-leading competitive swarm optimizer with dynamic Gaussian mutation for parameter extraction of solar photovoltaic models[J]. Energy Conversion and Management, 2020, 206:112450.

[12] ÁLVARO R L, LEONARDO V, MAURO C, et al. Multiobjective metaheuristic to design RNA sequences [J]. IEEE Transactions on Evolutionary Computation, 2019, 23(1): 156-169.

[13] 张庆磊. 基于种群排序策略的差分进化算法变异算子研究[D]. 泉州:华侨大学,2016.
ZHANG Qinglei. Research on differential evolution algorithm mutation operators based on population sorting strategy [D]. Quanzhou: Huaqiao University, 2016.

[14] LIU Zhenzu, WU Lianghong, ZHANG Hongqiang, et al. An improved competitive swarm optimizer for large scale optimization[M]. Singapore: Springer, 2020.

[15] MOHAPATRA P, DAS K N, ROY S. A modified competitive swarm optimizer for large scale optimization problems[J]. Applied Soft Computing, 2017, 59: 340-362.

[16] LING Tao, ZHAN Zhihui, WANG Yongxing, et al. Competitive swarm optimizer with dynamic grouping for large scale optimization[C]//2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 2018, 2655-2660.

[17] ZHOU Jianhong, FANG Wei, WU Xiaojun, et al. An opposition-based learning competitive particle swarm optimizer[C]//2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada, 2016, 515-521.

[18] DENG Hanbo, PENG Lizhi, ZHANG Haibo, et al. Ranking-based biased learning swarm optimizer for large-scale optimization [J]. Information Sciences, 2019, 493:120-137.

(责任编辑 周 蓓)