一种基于竞争性拥挤小生境技术的多峰优化算法

周新宇[1),2)] 田龙辉[1)] 明 飞[3)] 龚文引[3)] 王 晖[4)]

[1)](江西师范大学数字产业学院 江西 上饶 334000)

[2)](江西师范大学计算机信息工程学院 南昌 330022)

[3)](中国地质大学(武汉)计算机学院 武汉 430074)

[4)](江西水利电力大学信息工程学院 南昌 330099)

摘 要 在实际生产生活中,很多最优化问题通常有多个全局最优解,这类问题被称作多峰优化问题。拥挤小生境技术是求解多峰优化问题的一种有效手段,但因其仅考虑子代个体与父代个体之间的相似性,从而容易导致“替换错误”,造成部分全局最优解无法被定位。此外,在多峰优化问题中,部分全局峰值周围往往存在较为崎岖的地形,使得算法容易陷入局部最优。为此,本文提出了一种基于竞争性拥挤小生境技术的多峰优化算法。在经典的拥挤技术基础上设计了竞争性拥挤小生境技术,该技术不仅考虑子代个体和父代个体之间的相似性,还会同时考虑父代个体之间的相似性,将适应度较差的一个父代个体视作可替换个体;当子代个体与父代个体均不相似时,该可替换个体有机会被删除,以此减少替换错误的发生。并且,针对因崎岖地形而陷入局部最优的个体设计了一种虚拟小生境策略,通过在该个体周围生成若干个辅助个体来形成虚拟小生境,以对该个体做进一步搜索,从而提高找到全局峰值的可能性。为验证算法性能,在CEC2015测试集和2个实际优化问题(三角函数超越方程组、调频声波合成问题)上进行了大量实验验证,与11种代表性的多峰优化算法(包含2种CEC多峰优化竞赛的冠军算法)进行性能对比,结果表明:(1)在CEC2015测试集上,本文算法可在16个函数上取得最好结果,且在其中13个函数上能找到所有全局最优解,最好结果数量较表现最优的对比算法提升了15%;(2)在2个实际优化问题上,本文算法在峰值比指标上较表现最优的对比算法平均提升了12.05%。

关键词 多峰优化;拥挤技术;崎岖地形;局部搜索

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2026.00062

A Multimodal Optimization Algorithm Based on Competitive Crowding Niching Technique

ZHOU Xin-Yu[1),2)] TIAN Long-Hui[1)] MING Fei[3)] GONG Wen-Yin[3)] WANG Hui[4)]

[1)](School of Digital Industry, Jiangxi Normal University, Shangrao, Jiangxi 334000)

[2)](School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022)

[3)](School of Computer Science, China University of Geosciences (Wuhan), Wuhan 430074)

[4)](School of Information Engineering, Jiangxi University of Water Resources and Electric Power, Nanchang 330099)

Abstract In real-world production and daily life scenarios, many optimization problems typically possess multiple global optimal solutions. Such problems are referred to as multimodal optimization problems. The crowding niching technique has long been recognized as an effective method for solving multimodal optimization problems. However, a key limitation of this approach lies in its reliance solely on the similarity between offspring individuals and their parent individuals. This limitation can easily lead to “replacement errors”, and as a result, some global optima can-

收稿日期:2024-11-13;在线发布日期:2025-07-15。本课题得到国家自然科学基金(62366022,62166027)、江西省自然科学基金杰出青年基金(20212ACB212004)、江西省自然科学基金(20232BAB202048)资助。**周新宇**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为智能计算。E-mail:xyzhou@jxnu.edu.cn。**田龙辉**,硕士研究生,主要研究领域为智能优化算法。**明 飞**,博士研究生,主要研究领域为智能计算、进化多任务优化。**龚文引**,博士,教授,主要研究领域为智能计算、多目标优化等。**王晖(通信作者)**,博士,教授,主要研究领域为智能计算、多峰优化等。E-mail:huiwang@whu.edu.cn。

not be accurately located by the algorithm. Furthermore, for multimodal optimization problems, some global peaks are often surrounded by rugged landscapes. This characteristic increases the risk of the algorithm becoming trapped in local optima, thereby reducing its overall performance in optimizing multimodal optimization problems. To effectively address these issues, this paper proposes a novel multimodal optimization algorithm by designing a competitive crowding niching technique. Based on the classical crowding technique, the proposed competitive crowding niching technique incorporates a more comprehensive similarity assessment. Specifically, in addition to evaluating the similarity between offspring individuals and their parent individuals, the proposed technique also takes into account the similarity among the parent individuals themselves, and the parent individual with the poorer fitness value is considered as a potentially replaceable individual. If an offspring individual is dissimilar to all parent individuals, the potentially replaceable individual has a chance to be removed from the population. In this way, the algorithm can significantly reduce the likelihood of replacement errors through maintaining better population diversity. Moreover, some individuals may be trapped into local optima due to the rugged landscape, a virtual niching strategy is designed to overcome this issue. In this virtual niching strategy, some auxiliary individuals are generated within the neighborhood of trapped individual to form a virtual niche, aiming to further explore the surrounding search space of the trapped individual. This strategy is beneficial to increasing the chances of discovering the global peaks. Extensive experimental evaluations are conducted on the well-established CEC2015 multimodal test suite as well as on two real-world optimization problems (the trigonometric transcendental equation system and frequency-modulated sound wave synthesis problem). The proposed algorithm is compared against 11 state-of-the-art multimodal optimization algorithms, including two champion algorithms from previous CEC multimodal optimization competitions. The results demonstrate that: 1) On the CEC2015 test suite, the proposed algorithm achieves the best performance on 16 functions and successfully locates all global optima on 13 of them. The number of best results obtained by the proposed algorithm exceeds that of the best-performing comparison algorithm by 15%. 2) On the two real-world optimization problems, the proposed algorithm achieves an average improvement of 12.05% in terms of the Peak Ratio (PR) metric compared to the best-performing comparison algorithm.

Keywords　multimodal optimization; crowding technique; rugged landscape; local search

1 引　言

在实际的生产生活中,很多最优化问题通常有多个全局最优解,比如:多椭圆检测[1]、蛋白质结构预测[2]、桁架结构优化[3]以及特征选择[4]等问题。这类问题被称为多峰优化问题(Multimodal Optimization Problem,MMOP),而用于求解这类问题的算法被称作多峰优化算法[5-6]。对多峰优化算法而言,算法应有保持种群多样性的能力,以便找到多个全局最优解。近年来,进化算法(Evolutionary Algorithms,EAs)在最优化领域有出色表现,越来越多的研究人员在设计多峰优化算法时开始尝试EAs,且取得了较好求解效果。常见的 EAs 包括差分进化算法(Differential Evolution,DE)[7]、遗传算法(Genetic Algorithm,GA)[8]、人工蜂群算法(Artificial Bee Colony,ABC)[9]以及粒子群优化算法(Particle Swarm Optimization,PSO)[10]等。然而,因 EAs 最初是为求解单峰优化问题而设计的,缺少相关的种群多样性保持机制,使得 EAs 无法高效求解 MMOP。

为此,研究人员提出了小生境(Niching)技术来保持种群多样性。通过结合小生境技术,EAs 可更好地保持种群多样性,从而能高效求解 MMOP。小生境是生态学中的术语,指物种与特定环境之间的相互匹配,描述了物种对环境中的相关资源条件做

出的反应。小生境技术的主要思路是把种群划分为多个部分,使得不同部分的个体能在不同环境中独立进化,以避免个体趋同,提高算法找到多个全局最优解的可能性。目前,主流的小生境技术包括拥挤(Crowding)技术[11]、物种生成(Speciation)技术[12]、适应度共享(Fitness Sharing)技术[13]以及聚类(Clustering)技术[14]。在这些小生境技术中,拥挤技术因操作简单,所需参数少,受到了广泛关注,其主要操作方式是:从父代种群中随机抽取 CF 个个体,再从中找出与当前子代个体最相似的父代个体进行比较,保留适应度值更好的一个进入下一代。相似度一般以欧氏距离来决定,距离越近则相似度越高。拥挤技术的这种方式能减少个体间的拥挤程度,增加种群多样性,其中参数 CF 被称为拥挤因子,大小一般设置为 2 或 3。

然而,因 CF 取值较小,拥挤技术容易出现"替换错误(Replacement Errors)",即:随机选取的 CF 个父代个体和子代个体均不相似,但子代个体又必须替换其中一个不相似的父代个体,导致位于不同峰上的个体被误删。通常,越不相似的个体越有可能位于不同峰上。替换错误容易导致不同峰上的个体丢失,使得部分全局最优解无法被定位。为此,Thomsen 在提出的 CDE(Crowding Differential Evolution)算法中对 CF 值进行了调整[11],将其设置为整个种群大小,这样可在一定程度上减少替换错误。事实上,CDE 中的拥挤技术被公认是经典版本,之后采用了这一技术的相关多峰优化算法也几乎是基于这一版本[15-17]。但应指出的是,经典版本的拥挤技术依然存在替换错误,一个重要原因是并未考虑到父代个体自身的相似性,比如:当子代个体与所有父代个体均不相似,而父代个体自身之间又存在高相似度的个体,此时应避免子代个体和父代个体之间的替换,反而应在有高相似度的父代个体之间进行替换,以减少替换错误。

为此,本文提出了基于竞争性拥挤小生境技术的多峰优化算法,简称 CCDE-VN。在该算法中设计了一种竞争性拥挤(Competitive Crowding)技术,通过在经典拥挤技术的基础上增加考虑父代个体自身的相似性,使得在划分小生境时不仅考虑子代个体和父代个体的相似性,还进一步考虑父代个体之间的相似性。同时,针对 MMOP 中因崎岖地形①而导致开采难度较大的峰[19],设计了一种虚拟小生境(Virtual Niching,VN)策略,以生成辅助个体的方式来形成虚拟小生境,提高算法在这类峰上的收敛精度。为验证 CCDE-VN 的性能,在 CEC2015 多峰优化测试集上进行了大量实验,与 11 种知名多峰优化算法进行了性能对比(包括 CEC 多峰优化竞赛中的 2 个冠军算法),实验结果表明 CCDE-VN 具有更出色的性能。本文主要贡献可概括如下:

(1)设计了一种竞争性拥挤技术。不同于经典的拥挤技术只考虑子代个体和父代个体之间的相似性,该技术还同时考虑了父代个体自身的相似性,以此来进一步减小替换错误发生的可能性,提高算法找到更多全局最优解的可能性。

(2)设计了一种虚拟小生境策略。因 MMOP 中不同类型的峰值所处地形的特点一般不同,导致位于崎岖地形处的峰值很难开采。为此,虚拟小生境策略会在这类峰值的邻近个体处生成若干辅助个体,以形成一个虚拟小生境来提升峰的开采精度。相比于高斯扰动等经典局部搜索策略,该策略能显著提高找到全局最优解的能力。

本文其他部分安排如下:第 2 节简介相关工作,第 3 节详细介绍本文提出的 CCDE-VN 算法,第 4 节是实验部分,第 5 节对本文工作进行总结,并展望了未来工作。

2 相关工作

2.1 DE 算法

DE 是一种基于种群的启发式随机搜索算法[20],被广泛用于求解最优化问题,主要包括初始化、变异、交叉、选择四项操作。

(1)初始化操作

在算法开始时,先生成一个初始种群。假设种群有 NP 个个体,$X_i=(x_{i,1},x_{i,2},\cdots,x_{i,D})$ 表示第 i 个个体,$i\in\{1,2,\cdots,NP\}$,D 为问题维度。X_i 可用如下公式生成:

$$x_{i,j}=L_j+rand(0,1)\cdot(U_j-L_j) \quad (1)$$

其中,$x_{i,j}\in[L_j,U_j]$,U_j 和 L_j 分别表示第 j 维决策变量的上限和下限。$rand(0,1)$ 是[0,1]区间内均匀分布的随机数。

(2)变异操作

在初始化操作后,种群中的每个个体 X_i 都将执行变异操作,经典的变异操作有以下几种:

① 崎岖地形(Rugged landscape)[18]是指在优化问题的适应度函数搜索空间中的部分区域存在多个局部最优解,增加了算法搜索全局最优解的难度。

a)DE/rand/1:

$$V_i = X_{r1} + F \cdot (X_{r2} - X_{r3}) \tag{2}$$

b)DE/best/1:

$$V_i = X_{\text{best}} + F \cdot (X_{r1} - X_{r2}) \tag{3}$$

c)DE/current-to-best/1:

$$V_i = X_i + F \cdot (X_{\text{best}} - X_i) + F \cdot (X_{r1} - X_{r2}) \tag{4}$$

d)DE/rand/2:

$$V_i = X_{r1} + F \cdot (X_{r1} - X_{r2}) + F \cdot (X_{r3} - X_{r4}) \tag{5}$$

e)DE/best/2:

$$V_i = X_{best} + F \cdot (X_{r1} - X_{r2}) + F \cdot (X_{r3} - X_{r4}) \tag{6}$$

其中,$V_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,D})$ 为变异个体。参数 F 是缩放因子,用于控制差分向量大小,取值一般为 0 到 1 之间。$X_{r1}, X_{r2}, X_{r3}, X_{r4}$ 和 X_{r5} 均为种群中随机选取的互不相同的个体,且与 X_i 也不相同。X_{best} 是种群中适应度值最好的个体。

(3)交叉操作

在变异操作后,采用交叉操作生成 X_i 和 V_i 的试验个体 $U_i = (u_{i,1}, u_{i,2}, \cdots, u_{i,D})$。常用的二项式交叉操作如下:

$$u_{i,j} = \begin{cases} v_{i,j}, \text{if } rand(0,1) \leqslant CR \text{ or } j = j_{rand} \\ x_{i,j}, \text{otherwise} \end{cases} \tag{7}$$

其中,CR 为交叉率,取值一般为 0 到 1 之间。$rand(0,1)$ 是[0,1]区间内均匀分布的随机数。$j_{rand} \in \{1,2,\cdots,D\}$ 是随机选取的一个维度,以确保 U_i 至少有一维来自 V_i。

(4)选择操作

生成试验个体后,选择操作会在 X_i 和 U_i 中选择适应度值更好的一个进入下一代,该操作可表示为

$$X'_i = \begin{cases} U_i, \text{if } fit(U_i) > fit(X_i) \\ X_i, \text{otherwise} \end{cases} \tag{8}$$

其中 $X'_i = (x'_{i,1}, x'_{i,2}, \cdots, x'_{i,D})$ 是选择的下一代个体,$fit(\cdot)$ 表示适应度函数。

2.2　多峰优化算法的相关工作

设计多峰优化算法主要涉及两个方面:(1)如何保持种群多样性,以找到更多的全局最优解;(2)如何提高算法收敛精度,使得找到的全局最优解能满足精度要求。针对这两个方面,可将现有相关工作划分为三类:(1)应用小生境技术的多峰优化算法;(2)应用多目标技术的多峰优化算法;(3)应用局部搜索策略的多峰优化算法。前两类主要围绕如何保持种群多样性,第三类重点解决如何提高收敛精度。下面简介这三类相关工作:

(1)应用小生境技术的多峰优化算法

小生境技术是保持种群多样性的一种有效手段,近年来如何应用小生境技术来设计多峰优化算法是求解 MMOP 的一个研究热点。例如,Li 等人[12]提出了一种应用物种生成技术的多峰优化算法(简称 SDE),先将种群中适应度较好的个体视作小生境中心,再把与该中心同处指定半径内的个体划分至同一小生境,使得 SDE 有较强开采能力。基于 CDE 和 SDE,Gao 等人[14]结合聚类技术提出了知名的 Self-CCDE 和 Self-CSDE 算法。在 Self-CCDE 中,通过聚类技术对随机个体的选取进行限定,从而增强算法收敛性;对于 Self-CSDE,则通过聚类技术避免了小生境半径的设置问题,进一步优化了算法流程。Lin 等人[21]提出了一种应用 Nearest-Better(NB)聚类的多峰优化算法(简称 FBK-DE),采用了改进的 NB 聚类来划分小生境,并设计了物种平衡策略用于动态调整小生境大小,有效避免了大量个体聚集于同一小生境的问题。Jiang[22]在提出的 NCD-DE 算法中设计了一种新的小生境中心选择方法,将小生境中心的选择问题转化为 0-1 编码的离散优化问题,再通过 GA 算法来优化小生境中心,从而避免小生境的参数设置。与 NCD-DE 算法类似,Liang[23]等人针对小生境中心的设置问题提出了 NCIDE 算法,通过同时考虑适应度和距离来动态识别小生境中心,并将非中心个体归入最近小生境。类似地,为缓解小生境设置的参数敏感问题,Wang 等人[24]提出了一种自适应小生境规模的多峰优化算法(简称 AED-DDE),通过单变量高斯分布来拟合种群的个体分布,将符合同一分布的个体划分至同一小生境,并且采用主从小生境分布模型用于弥补单个小生境与整个种群存在交互不足的问题。Zhao 等人[25]提出了一种应用 Local Binary Pattern(LBP)小生境技术的多峰优化算法(简称 LBPADE),模拟图像处理过程中的 LBP 算子为每个个体构建小生境,以更好地定位更多的全局最优解。最近,Zhou[26]等人提出了一种融合物种生成技术和拥挤技术的混合小生境策略,采用适应度地形分析技术将种群划分为两部分,对适应度较好部分应用物种生成技术,而较差部分则使用拥挤技术,以更好平衡算法的收敛性和多样性。

(2)应用多目标技术的多峰优化算法

除小生境技术外,还有一些相关工作将 MMOP

转化为多目标优化问题(MOP)进行求解。通常,该类工作会将 MMOP 的适应度函数作为 MOP 的第一个目标,而第二个目标的设计则因算法而异。例如,Cheng 等人[27]将自适应多样性指标作为第二个目标,该指标会随进化代数增加而动态调整,以实现开采和多样性的动态平衡。Basak 等人[28]在提出的 MOBiDE 算法中,将当前个体与其他所有个体的平均距离作为第二目标,再结合非支配排序与超体积度量排序,有效减少了个体聚集情况,进一步增强了种群多样性。类似地,Bandaru 等人[29]将当前个体与其他所有个体的距离平方和的倒数作为第二目标,在 NSGA-Ⅱ框架中引入了一种无参数的小生境技术来调整支配准则,使得当两个个体能满足一定距离要求时才能使用支配关系。与上述三种算法不同的是,Wang 等人[30]在提出的 MOMMOP 算法中将两个优化目标都进行了重新定义,第一个目标定义为 $x_1+a(X)$,而第二个目标为 $1-x_1+a(X)$,x_1 为个体在第一个维度上的变量值,$a(X)$ 是与适应度相关的函数,该方式可确保两个目标相互冲突,更好地保持多样性,以找到更多全局最优解。

(3)应用局部搜索策略的多峰优化算法

为提高算法收敛精度,这一类相关工作重点考虑如何应用局部搜索策略。例如,Wang 等人[31]提出了一种应用简化协方差矩阵自适应演化策略(MAES)的多峰优化算法,与协方差矩阵自适应演化策略(CMA-ES)相比,MAES 不再计算协方差矩阵,降低了算法计算复杂度。Wang 等人[32]在提出的 FDLS-ADE 算法中,将局部搜索范围与进化过程相关联,采用了一种标准差可动态变化的高斯扰动策略,通过逐步减小标准差,使算法在早期能以较大范围进行搜索,增强全局勘探能力,而后期则聚焦于小范围的精细搜索,可有效平衡勘探与开采能力。类似地,Chen 等人[19]提出的算法也利用了高斯扰动策略,但标准差的变化方式有所不同,其变化情况与个体变化情况相关,当一个个体若干次未成功更新时,便缩小采样区域以提高开采能力。根据小生境中心的优劣情况,Sheng 等人[33]提出了一种应用自适应柯西搜索策略的多峰优化算法,对适应度较差的小生境中心,会以较大的步长进行扰动,使算法能快速接近全局最优解;而对适应度较好的小生境中心,则会以较小步长进行扰动,从而提高算法收敛精度。Jiang 等人[22]在提出的 NCD-DE 算法中,引入了宽、窄两种不同范围的局部搜索方程,其中宽范围的搜索方程有助于个体跳出局部最优解,而窄范围的搜索方程可进一步提升算法的收敛精度。Zhao[16]等人在提出的 OADE 算法中,将搜索方程与问题维度相结合,当问题维度较低时,搜索方程以当前个体来引导变异,而当维度较高时,则以小生境中最优个体来引导变异,提高了算法性能。

3 本文算法

本文提出的 CCDE-VN 算法主要包含两个阶段:1)在第一个阶段中设计了竞争性拥挤小生境技术和自适应高斯扰动策略,目的是定位更多的全局最优解并提升其精度;2)第二阶段使用了虚拟小生境策略,对因崎岖地形陷入局部峰值的个体做进一步搜索,以定位到附近的全局峰值。下面详细介绍相关内容。

3.1 竞争性拥挤小生境技术

在经典的拥挤技术中,若子代个体的适应度优于最相似(欧氏距离最近)的父代个体,那么将替换该父代个体,以维持种群多样性。然而,当子代个体与所有父代个体均不相似时,这会导致替换错误。为方便说明,在图 1 中给出了一个示例,红色点代表子代个体,蓝色点代表父代个体,黑色五角星是全局最优解。从图中可看出距子代个体 X_i^c 最近的父代个体为 X_n^p,这两者位于不同峰上。如果采用经典拥挤技术,X_n^p 会因适应度值较差而被替换,从而导致全局最优解 O_b 周围缺少个体,使得算法难以定位到 O_b。

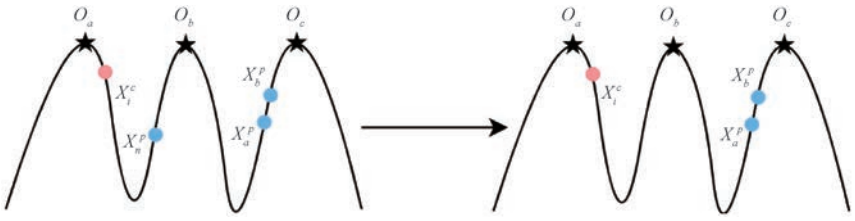


图 1 经典拥挤技术示意图

为此,本文在经典拥挤技术的基础上设计了竞争性拥挤技术,从父代种群中找到两个最相似的父代个体,将其中适应度较差的一个作为新增的可替换个体,使得子代个体与所有父代个体均不相似时,该新增的可替换个体有机会被删除,以减少替换错误。以图 2 为例做进一步说明,图中 X_i^c 为子代个体,X_n^p 是与 X_i^c 最相似的父代个体,这两者距离为 S_{cp}。X_a^p 与 X_b^p 是父代种群中最相似的两个个体,且假设 X_a^p 的适应度差于 X_b^p,两者的距离为 S_{pp}。按 S_{cp} 与 S_{pp} 的大小关系可分为以下两种情况:

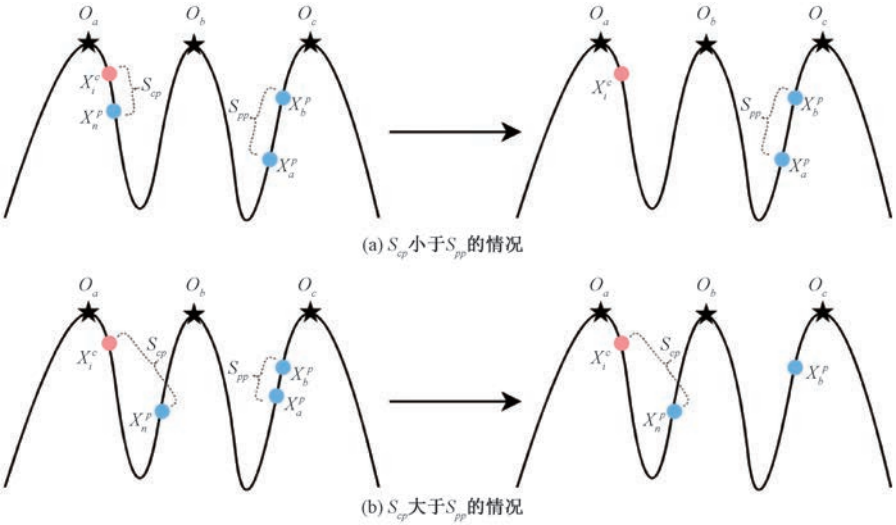


图 2　竞争性拥挤技术示意图

(1) S_{cp} 小于 S_{pp},如图 2(a)所示。子代个体与最近的父代个体相似度较高,若子代个体适应度更好,可直接替换最近的父代个体。图中 X_i^c 优于 X_n^p,此时 X_n^p 被替换。

(2) S_{cp} 大于 S_{pp},如图 2(b)所示,此时与图 1 情况相同。此时子代个体与最近的父代个体相似度要低于父代种群中最相似的两个个体,即 X_i^c 和 X_n^p 的相似度要低于 X_a^p 和 X_b^p,在竞争性拥挤技术中会先比较 X_a^p 和 X_b^p,再将两者中适应度较差的一个与 X_i^c 相比,这点不同于经典拥挤技术。图中 X_i^c 优于 X_a^p,表明 X_a^p 不仅拥挤度较高且适应度也较差,此时应被替换。需注意的是,X_i^c 也可能差于 X_a^p,但因 X_i^c 和 X_n^p 可能位于同一峰值,X_i^c 不应被直接舍弃,而应与 X_n^p 再做进一步比较,以保留两者中适应度较好的一个。

此外,在竞争性拥挤技术中还存在一种特殊情况:当父代种群中最相似的两个个体适应度都接近最优值时,子代个体将很难有机会替换掉其中较差的一个。此时,算法只能选择替换与该子代个体最近的父代个体,这也会增加替换错误的可能性。为此,本文还设计了一种检测机制:在每次进行选择操作时,检测父代种群中最相似的两个个体中较差的一个是否被成功替换,若连续 φ 次未能成功替换,在这种情况下不宜再比较适应度,而应直接用子代个体替换其中较差的父代个体。本文实验中参数 φ 设为 $30 \cdot D$,D 表示问题维度,关于该参数的敏感性分析在实验部分给出了。

为更好地说明竞争性拥挤技术,在算法 1 中给出了伪代码表示,其中 NP 为种群大小,D 为问题维度,F 为缩放因子,CR 为交叉概率,φ 为阈值参数,FEs 为评估次数,pop 为种群,φCnt 为计数器,$fit(\cdot)$ 为适应度函数。

算法 1. 竞争性拥挤小生境技术

输入:NP,D,$F=0.1$,$CR=0.3$,φ,FEs,pop,$\varphi Cnt=0$;

输出:pop,FEs,φCnt;

1. FOR $i=1$ TO NP DO
2. 　用 DE/rand/1 生成子代个体 X_i^c;
3. 　计算 X_i^c 适应度,$FEs=FEs+1$;
4. 　在 pop 中找到与 X_i^c 最相似的个体 X_n^p,两者距离记为 S_{cp};
5. 　在 pop 中找到最相似的两个个体 X_a^p 和 X_b^p,两者距离记为 S_{pp};
6. 　将 X_a^p 和 X_b^p 中适应度较差的个体记为 X_w^p;
7. 　检测 X_w^p 是否被成功替换过,若被成功替换过,则令 $\varphi Cn=0$;否则,令 $\varphi Cn=\varphi Cnt+1$;
8. 　IF $\varphi Cnt \geqslant \varphi$ THEN
9. 　　使用 X_i^c 替换 X_w^p,$\varphi Cnt=0$, break;
10. 　END IF
11. 　IF $S_{cp} < S_{pp}$ THEN

IF $fit(X_i^c) > fit(X_n^p)$ THEN
使用 X_i^c 替换 X_n^p；
END IF
ELSE
IF $fit(X_i^c) > fit(X_w^p)$ THEN
使用 X_i^c 替换 X_w^p；
ELSEIF $fit(X_i^c) > fit(X_n^p)$ THEN
使用 X_i^c 替换 X_n^p；
END IF
END IF
END FOR

3.2 自适应高斯扰动策略

在多峰优化算法中，算法的收敛精度描述了算法所得解与问题的全局最优解之间的接近程度。收敛精度越高表明算法的求解质量越高，越容易达到精度要求。通常，局部搜索策略是提高算法收敛精度的一种常见手段，例如前文 2.2 节“相关工作”中所述的第三类相关工作。高斯扰动策略因操作简单、计算复杂度低而受到广泛应用。标准的高斯扰动策略如下式所示：

$$X_i' = Gaussian(X_i, \sigma) \tag{9}$$

其中，X_i' 是高斯扰动后生成的新个体，X_i 为当前个体，σ 是标准差。一般来说，σ 越大，扰动幅度越大，σ 越小，扰动幅度越小。

通常，算法在初期应注重勘探能力，而在后期则需偏重开采能力。因此，应用了标准高斯扰动策略的相关算法一般会将 σ 值设置为在进化过程中逐渐递减。但应指出的是，这种方式并没有考虑到个体差异性，因为适应度不同的个体所适合的扰动范围也往往不同。适应度较差个体需较大 σ，有助于跳出局部最优；而适应度较好个体则适合较小 σ，以提高收敛精度。为此，本文提出自适应高斯扰动策略，综合考虑进化过程和个体差异性，为每个个体设定自适应 σ 值，具体如下式(10)和(11)所示：

$$X_i' = Gaussian(X_i, \sigma_i) \tag{10}$$

$$\sigma_i = \min\left[\max\left(\frac{fit(X_{\text{best}}) - fit(X_i)}{fit(X_{\text{best}}) - fit(X_{\text{worst}}) + 1.0E-10}, lb\right), ub\right] \tag{11}$$

其中，$fit(\cdot)$ 表示适应度函数，X_{best} 和 X_{worst} 分别表示种群中适应度最好和最差的个体，X_i 为当前个体。可看出当 X_i 的适应度越好时，σ_i 越小；相反，当 X_i 适应度越差，σ_i 越大。同时，max 函数可将 lb 置为下界，min 函数将 ub 置为上界。ub 和 lb 会随着评估次数 FEs 的增加而减小，其计算公式如下所示：

$$lb = 10 \wedge \left(\frac{-10 \cdot FEs}{MaxFEs}\right) \tag{12}$$

$$ub = 10 \wedge \left(\frac{-10 \cdot FEs}{MaxFEs} + 4\right) \tag{13}$$

在图 3 中给出了 ub 与 lb 的变化趋势情况。从图中可看出 σ_i 在进化过程中会逐步减小，使得 σ_i 不仅与个体适应度相关，同时也与进化过程动态关联，进一步提升了高斯扰动策略的灵活度与有效性。自适应高斯扰动策略的伪代码如算法 2 所示。

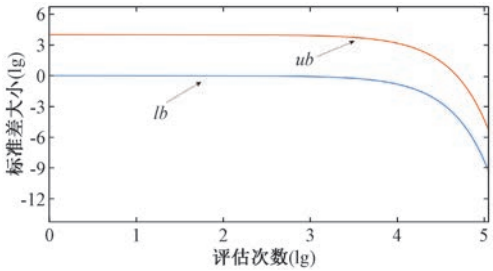


图 3 ub 与 lb 变化趋势

算法 2. 自适应高斯扰动策略

输入：NP，FEs，pop；

输出：pop，FEs；

FOR $i=1$ TO NP DO
按自适应高斯扰动公式，即式(10)，生成新个体 X_i'；
计算 X_i' 适应度值，$FEs = FEs + 1$；
IF $fit(X_i') > fit(X_i)$ THEN
使用 X_i' 替换 X_i；
END IF
END FOR

3.3 虚拟小生境策略

多峰优化问题中通常存在一些较为崎岖的地形，这种地形一般包含多个局部峰值，若全局峰值（即全局最优解）位于这种崎岖地形中，这对算法的求解会带来较大挑战。以图 4 为例，O_a 和 O_b 分别代表优化问题的两个全局峰值，可看出 O_a 周围的地形相对平滑，没有明显的局部峰值；相比之下，O_b 周围则较为崎岖，有较多的局部峰值；因此，算法定位到 O_b 的难度更大，容易陷入局部最优。为此，

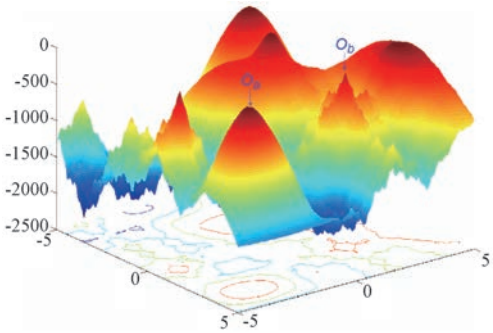


图 4 崎岖地形示意图

针对存在类似 O_b 这种全局峰值的 MMOP,本文专门设计了一种虚拟小生境策略,以对此类峰值附近的个体做进一步搜索,帮助其跳出局部最优,提高定位到全局峰值的可能性。

虚拟小生境策略的主要思路是对峰值附近的个体做进一步的细粒度搜索,然而峰值附近可能会存在多个个体,若所有个体均搜索,会浪费适应度函数评估次数。同时,一般情况下一个峰值仅对应一个个体,也无需搜索所有附近个体。因此,在执行虚拟小生境策略之前,本文先采用清除技术[34]处理种群,使得同一峰值处仅保留一个适应度最好的个体,清除其余非必要个体。清除技术最早由 Holland 提出[31],主要思路是先将种群中的所有个体按适应度从好到差进行排序,选择适应度最好的个体作为中心,再清除该中心指定半径范围内的其他个体;之后选择下一个适应度最好且未被清除的个体作为中心,重复以上步骤,直至所有个体要么被选中保留,要么被清除出种群。通常,清除半径 r 设定为 $0.05\cdot D$,D 为问题维度。为更好地说明清除技术,在算法 3 中给出了其伪代码表示。

算法 3. 清除技术

输入:pop,D;

输出:pop;

1. 将 pop 中所有个体按适应度从好到差进行排序;
2. 设定清除半径 $r=0.05\cdot D$;
3. FOR $i=1$ TO $|pop|-1$ DO
4. FOR $j=i+1$ TO $|pop|$ DO
5. 若 X_i 与 X_j 距离小于 r,将 X_j 从 pop 中清除;
6. END FOR
7. END FOR

在完成种群处理后,将其他个体与最好个体的适应度差值作为判断标准,用于评估这些个体是否达到精度要求,以对未达到精度要求的个体再使用虚拟小生境策略。值得注意的是,由于无法确定适应度最好的个体自身是否满足精度要求,因此会直接采用虚拟小生境策略来处理该最好个体,之后再依次比较其他个体与最好个体的适应度差值。当差值小于 1.0E−6,则认定该个体已经达到精度要求,无需再用虚拟小生境策略;相反,若大于 1.0E−6,则表明该个体可能处于崎岖地形,陷入了局部最优,需采用虚拟小生境策略来进一步搜索。对需要被处理的个体而言,先计算该个体与种群中最近个体的距离 d,再以该个体为中心,在半径为 $d/2$ 的范围内生成 $k-1$ 个辅助个体来形成虚拟小生境。辅助个体的生成方式如下所示:

$$x'_{z,j}=x_{i,j}+[rand(0,1)\cdot 2-1]\cdot d/2 \tag{14}$$

其中 $X'_z=(x_{z,1},x_{z,2},\cdots,x_{z,D})$ 表示辅助个体,$z\in\{1,2,\cdots,k-1\}$,$X_i=(x_{i,1},x_{i,2},\cdots,x_{i,D})$ 为被处理个体,$rand(0,1)$ 是[0, 1]区间内均匀分布的随机数。为避免小生境范围过大,若 d 值大于 1,则将其限制为 1。$k=\min(60,15\cdot D)$,其中 D 为问题维度。

在形成虚拟小生境之后,将采用经典的 DE/rand/1 策略对这 k 个个体进行处理,以充分搜索被处理个体的周围空间,帮助其跳出局部最优。当虚拟小生境中最好个体和最差个体的适应度差值小于 1.0E−6 时,则认定该虚拟小生境已收敛,算法将继续判断下一个个体是否需使用虚拟小生境策略。以图 5 为例做进一步说明,图中蓝色点代表种群中个体,同一虚线圆框中的粉色点为辅助个体。假设 X_a 是种群中最好个体,但因无法确定其是否达到精度要求,所以会直接采用虚拟小生境策略进行搜索。之后,继续处理 X_b,若其与 X_a 的适应度差值小于 1.0E−6,则不再处理;否则,将对 X_b 采用虚拟小生境策略。图中个体 X_b 位于局部峰值,与 X_a 的适应度差距较大,因此会对其使用虚拟小生境策略。

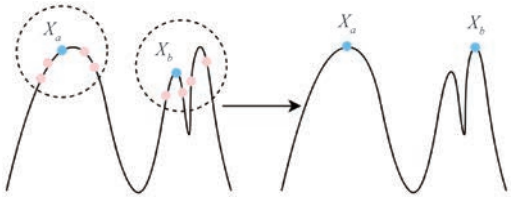


图 5　虚拟小生境技术示意图

需说明的是,因无法确定每个陷入局部峰值的个体附近是否存在全局峰值,如果将大量的评估次数用于虚拟小生境策略,可能会在远离全局峰值的个体上使用虚拟小生境策略,从而浪费适应度函数评估次数。因此,为控制虚拟小生境策略的计算开销,本文将算法划分为两个阶段,算法前 80%的适应度函数评估次数用于第一阶段,执行竞争性拥挤技术和自适应高斯扰动策略;剩余的 20%用于第二阶段,执行虚拟小生境策略。为更好地说明虚拟小生境策略,在算法 4 中给出了伪代码表示。

算法 4. 虚拟小生境策略

输入:pop, D, FEs, $MaxFEs$, $F=0.5$, $CR=0.9$, $\varepsilon=1.0E-6$;

输出:pop;

1. FOR $i=1$ TO $|pop|$ DO

2. IF $FEs > MaxFEs$ THEN
3. break;
4. END IF
5. $S = \emptyset$;
6. IF $i == 1 \,||\, fit(X_1) - fit(X_i) > \varepsilon$ THEN
7. 计算 X_i 与种群中最近个体的距离 d;
8. 按式(14)生成 $k-1$ 个辅助个体;
9. 计算辅助个体适应度，$FEs = FEs + k - 1$;
10. 定义 S 用于保存 X_i 和 $k-1$ 个辅助个体;
11. WHILE $\max(fit(S)) - \min(fit(S)) > \varepsilon$
12. FOR $j = 1$ TO k DO
13. 将当前个体记为 X_j^S，在小生境中用 DE/rand/1 生成试验个体 U_j;
14. 计算 U_j 适应度，$FEs = FEs + 1$;
15. 如果 U_j 优于 X_j^S，则用 U_j 替换 X_j^S;
16. END FOR
17. END WHILE
18. END IF
19. END FOR

3.4 CCDE-VN 算法整体流程

为更清楚地说明本文提出的 CCDE-VN 算法，在算法 5 中给出了其伪代码，其中参数 η 把算法划分为两阶段用于分别执行相关策略，该参数的敏感性分析在实验部分中已给出。与其他使用拥挤小生境技术的多峰优化算法相比，CCDE-VN 主要有如下四点修改之处：

(1)第 3 行：将经典的拥挤技术改进为竞争性拥挤技术。

(2)第 4 行：将经典的高斯扰动策略改进为自适应高斯扰动策略。

(3)第 6 行：应用清除技术。

(4)第 7 行：提出了虚拟小生境策略。

算法 5. CCDE-VN

输入：η，FEs，$MaxFEs$;

输出：pop;

1. WHILE $FEs < MaxFEs$ THEN
2. IF $FEs < \eta \cdot MaxFEs$ THEN
3. 执行竞争性拥挤小生境技术(算法 1);
4. 执行自适应高斯扰动策略(算法 2);
5. ELSE
6. 执行清除技术(算法 3);
7. 执行虚拟小生境策略(算法 4);
8. break;
9. END IF
10. END WHILE

3.5 算法时间复杂度分析

因 CCDE-VN 算法的主要步骤涉及四个子算法，因此其时间复杂度也由这四个子算法决定。设算法的最大迭代次数为 G，种群大小是 NP，问题维度为 D。下面分别分析这四个子算法的时间复杂度：

(1)在算法 1 竞争性拥挤小生境技术中，需对所有个体进行变异交叉操作，这部分的时间复杂度为 $O(NP \cdot D)$。在父代种群找到距离最近的两个个体，可定义一个距离表来实现，相应的时间复杂度为 $O(NP^2 \cdot D)$。查找距离子代最近的父代个体为 $O(NP \cdot D)$。若种群更新，可将已算出的子代与父代距离存入表中，因此该部分的时间复杂度为 $O(1)$。第一阶段的迭代次数为 $0.8 \cdot G$，因此算法 1 的时间复杂度为 $O(0.8 \cdot G \cdot (NP \cdot D + NP^2 + NP \cdot D) + NP^2 \cdot D + 1)$。因 $NP > D$，且仅需生成一次距离表，所以算法 1 的最终时间复杂度可计为 $O(G \cdot NP^2)$。

(2)在算法 2 自适应高斯扰动策略中，需找到种群中适应度值最好与最差的个体，该过程的时间复杂度是 $O(NP)$。对所有个体使用高斯扰动的时间复杂度为 $O(NP \cdot D)$。选择操作是一对一选择，时间复杂度为 $O(NP)$。因此，算法 2 的时间复杂度为 $O(0.8 \cdot G \cdot (NP + NP \cdot D + NP))$。因 $D \geqslant 1$，所以算法 2 的最终时间复杂度为 $O(G \cdot NP \cdot D)$。

(3)在算法 3 清除策略中，主要操作是找到与当前个体在一定阈值内的其他个体，因此时间复杂度可计为 $O(NP^2 \cdot D)$。

(4)在算法 4 虚拟小生境策略中，每个虚拟小生境内有 $15 \cdot D$ 个个体进行 DE 操作，因此时间复杂度为 $O(15 \cdot D^2)$，第二阶段的迭代次数为 $0.2 \cdot G$，因此算法 4 的最终时间复杂度为 $O(G \cdot D^2)$。

综上，CCDE-VN 算法的时间复杂度为 $O(G \cdot NP^2 + G \cdot NP \cdot D + NP^2 \cdot D + G \cdot D^2)$。由于 $G > NP > D$，因此算法的最终时间复杂度可计为 $O(G \cdot NP^2)$。

4 实验验证和结果分析

4.1 测试函数和参数设置

为验证 CCDE-VN 算法性能，本文采用 CEC2015 测试集。需注意的是，该测试集与 CEC2013 测试集[35]完全相同，是多峰优化领域中应用最广泛的一套测试集。CEC2015 包含 20 个不同维度的测试函

数,均为求最大值,相关信息如表 1 所示。其中,F1-F5 是相对简单的多峰函数,F6-F10 为可扩展的多峰函数,而 F11-F20 是由多个基本函数组合形成的复杂多峰函数。此外,每个函数所对应的最大评估次数和种群规模均按测试集的原文献建议进行设置[35]。

表 1　CEC2015 测试集的相关情况

函数	全局最优解数量	问题维度	最大评估次数	建议的种群规模
F1	2	1	50000	80
F2	5	1	50000	80
F3	1	1	50000	80
F4	4	2	50000	80
F5	2	2	50000	80
F6	18	2	200000	100
F7	36	2	200000	300
F8	81	3	400000	300
F9	216	3	400000	300
F10	12	2	200000	100
F11	6	2	200000	200
F12	8	2	200000	200
F13	6	2	200000	200
F14	6	3	400000	200
F15	8	3	400000	200
F16	6	5	400000	200
F17	8	5	400000	200
F18	6	10	400000	200
F19	8	10	400000	200
F20	8	20	400000	200

为方便对比不同算法,本文采用了峰值率 PR(Peak Ratio)和成功率 SR(Success Rate)作为对比的性能指标。PR 是算法在多次运行中能找到的全局最优解个数占所有全局最优解总个数的平均比例,其计算公式如下:

$$PR=\frac{\sum_{i}^{NR}NPF_i}{NKP\cdot NR} \tag{15}$$

其中,NPF_i 是第 i 次运行中找到的全局最优解个数,NKP 是优化问题的所有全局最优解的总个数,NR 表示算法运行次数。第二个性能指标 SR 表示算法多次运行中成功运行次数与所有运行次数的比值,其中成功运行是指算法在一次运行中能找到所有全局最优解。SR 的计算公式如下所示,其中 NSR 表示成功的运行次数。

$$SR=\frac{NSR}{NR} \tag{16}$$

本文的参数设置为:种群大小 NP 按 CEC2015 测试集的建议进行设置(如表 1),阶段划分参数 η 为 0.8,竞争性拥挤技术中阈值 φ 为 $30\cdot D$,虚拟小生境策略的辅助个体数 $k=\min(60,15\cdot D)$。

4.2　与其他相关算法对比

本文选取了 11 种代表性的多峰优化算法进行性能对比,具体情况如表 2 所示。在对比算法中,算法 1～7 是应用小生境技术的多峰优化算法,其中算法 1～4 是应用经典小生境技术及其改进版本的算法,而算法 5～7 是近年来提出的应用新型小生境技术的算法。算法 8 是应用局部搜索的多峰优化算法,算法 9 是应用多目标技术的算法。值得说明的是,算法 10 和 11 分别是 CEC2013 和 CEC2015 多峰优化竞赛的冠军算法。关于这些算法的简介可参考本文第 2 节“相关工作”部分。

表 2　对比算法的相关情况

算法类别	算法名称	算法描述	算法年份
应用小生境技术的多峰优化算法	1. CDE[11]	应用拥挤小生境技术的多峰优化算法	2003 年
	2. SDE[12]	应用物种生成小生境技术的多峰优化算法	2005 年
	3. Self-CCDE[14]	应用聚类方法改进拥挤小生境技术的多峰优化算法	2014 年
	4. Self-CSDE[14]	应用聚类方法改进物种生成小生境技术的多峰优化算法	2014 年
	5. FBK-DE[21]	应用最近更好聚类划分小生境的多峰优化算法	2019 年
	6. NCD-DE[22]	应用 GA 算法优化小生境中心的多峰优化算法	2021 年
	7. AED-DDE[24]	应用高斯分布划分小生境的多峰优化算法	2020 年
应用局部搜索的多峰优化算法	8. FDLS-ADE[32]	应用高斯扰动进行局部搜索的多峰优化算法	2023 年
应用多目标技术的算法	9. MOMMOP[30]	应用多目标技术的多峰优化算法	2015 年
CEC 多峰优化竞赛的冠军算法	10. dADE/nrand/1[36]	应用动态存档技术的多峰优化算法	2013 年
	11. NMMSO[37]	应用动态多种群技术的多峰优化算法	2014 年

所有算法在每个测试函数上独立运行 51 次,取平均值作为最终结果,解的精度为 1.0E-4。此外,为更好地突出不同算法在结果上的差异性,当算法取得“1.000”结果时,用“1”进行表示;而对“0.000”结果用“0”表示。为方便起见,对每个测试函数的最优结果进行加粗凸显。需说明的是,运行次数设置为 51 次的原因有两方面:(1)这是多峰优化算法领域中的常见做法,例如:NCD-DE[22]、AED-DDE[24]、FDLS-ADE[32],因此本文采用相同设置,方便与这些相关算法进行对比;(2)奇数次实验会有一个明确

的中位数值，可避免偶数次实验中可能出现的中位数不明确和对称性偏差问题。

为客观对比，所有对比算法的实验结果均来源于公开的文献报道，具体为：(1)CDE 和 SDE 结果来源于文献[5]中表 3；(2)Self-CCDE、Self-CSDE、NMMSO 结果来源于文献[6]中表 2；(3)FBK-DE 结果来源于文献[21]中表 4；(4)NCD-DE 结果来源于文献[22]中表 1；(5)AED-DDE 结果来源于文献[24]中表 4；(6)FDLS-ADE 结果来源于文献[32]中表 4；(7)MOMMOP 结果来源于文献[16]中表 3；(8)dADE/nrand/1 结果来源于文献[36]中表 2。表 3 给出了详细的对比结果，其中“+”表示本文算法性能更好，“−”表示本文算法性能更差，“=”表示本文算法与对比算法性能相当。从表 3 中可看出，本文算法 CCDE-VN 在所有测试函数上均有较好表现，在 F1～F13、F15 以及 F17～F18 这 16 个函数上能取得最好结果，特别是在 F1～F13 上能找到所有的全局最优解，*PR* 和 *SR* 均为 1。具体的实验结果分析如下：

表 3 CCDE-VN 与其他 11 种多峰优化算法的对比结果

函数	CCDE-VN		CDE		SDE		Self-CCDE		Self-CSDE		FBK-DE	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**(=)	1	0.657(+)	0.373	**1**(=)	1	**1**(=)	1	**1**(=)	1
F2	**1**	1	**1**(=)	1	0.737(+)	0.529	**1**(=)	1	**1**(=)	1	**1**(=)	1
F3	**1**	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F4	**1**	1	**1**(=)	1	0.284(+)	0	**1**(=)	1	0.620(+)	0.160	**1**(=)	1
F5	**1**	1	**1**(=)	1	0.922(+)	0.843	**1**(=)	1	0.990(+)	0.980	**1**(=)	1
F6	**1**	1	**1**(=)	1	0.056(+)	0	0.953(+)	0.400	0.692(+)	0.080	0.990(+)	0.820
F7	**1**	1	0.861(+)	0	0.054(+)	0	0.648(+)	0	0.466(+)	0	0.813(+)	0
F8	**1**	1	0(+)	0	0.015(+)	0	0.678(+)	0	0.312(+)	0	0.824(+)	0
F9	**1**	1	0.474(+)	0	0.011(+)	0	0.260(+)	0	0.107(+)	0	0.425(+)	0
F10	**1**	1	**1**(=)	1	0.147(+)	0	**1**(=)	1	0.985(+)	0.860	**1**(=)	1
F11	**1**	1	0.330(+)	0	0.314(+)	0	0.743(+)	0.060	0.493(+)	0	**1**(=)	1
F12	**1**	1	0.002(+)	0	0.208(+)	0	0.275(+)	0	0.253(+)	0	0.935(+)	0.480
F13	**1**	1	0.141(+)	0	0.297(+)	0	0.623(+)	0	0.427(+)	0	**1**(=)	1
F14	0.846	0.255	0.026(+)	0	0.216(+)	0	0.657(+)	0	0.300(+)	0	**0.907**(−)	0.460
F15	**0.762**	0	0.005(+)	0	0.108(+)	0	0.318(+)	0	0.150(+)	0	0.730(+)	0
F16	0.680	0	0(+)	0	0.108(+)	0	0.587(+)	0	0.023(+)	0	**0.707**(−)	0
F17	**0.711**	0	0(+)	0	0.076(+)	0	0.248(+)	0	0.023(+)	0	0.630(+)	0
F18	**0.667**	0	0.167(+)	0	0.026(+)	0	0.290(+)	0	0(+)	0	**0.667**(=)	0
F19	0.515	0	0(+)	0	0.105(+)	0	0.088(+)	0	0(+)	0	**0.520**(−)	0
F20	0.265	0	0(+)	0	0(+)	0	0.070(+)	0	0(+)	0	**0.450**(−)	0
最好个数	**16**		7		1		6		3		13	

函数	NCD-DE		AED-DDE		FDLS-ADE		MOMMOP		dADE/nrand/1		NMMSO	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F2	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F3	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F4	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F5	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F6	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	0.984(+)	0.780	0.993(+)	0.880
F7	0.905(+)	0.078	0.838(+)	0.039	0.951(+)	0.216	**1**(=)	1	0.823(+)	0	**1**(=)	1
F8	0.961(+)	0.098	0.747(+)	0	0.999(+)	0.961	**1**(=)	1	0.967(+)	0.140	0.900(+)	0
F9	0.553(+)	0	0.384(+)	0	0.572(+)	0	**1**(=)	1	0.431(+)	0	0.978(+)	0.100
F10	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1	**1**(=)	1
F11	**1**(=)	1	**1**(=)	1	**1**(=)	1	0.716(+)	0.020	0.667(+)	0	0.997(+)	0.980
F12	0.993(+)	0.941	**1**(=)	1	**1**(=)	1	0.939(+)	0.549	0.740(+)	0	0.985(+)	0.880
F13	0.892(+)	0.412	0.686(+)	0	0.702(+)	0.020	0.667(+)	0	0.667(+)	0	0.997(+)	0.980
F14	0.683(+)	0	0.667(+)	0	0.667(+)	0	0.667(+)	0	0.667(+)	0	0.707(+)	0
F15	0.640(+)	0	0.637(+)	0	0.689(+)	0	0.618(+)	0	0.627(+)	0	0.653(+)	0
F16	0.667(+)	0	0.667(+)	0	0.667(+)	0	0.650(+)	0	0.667(+)	0	0.660(+)	0
F17	0.522(+)	0	0.375(+)	0	0.446(+)	0	0.505(+)	0	0.403(+)	0	0.468(+)	0
F18	**0.667**(=)	0	0.654(+)	0	**0.667**(=)	0	0.497(+)	0	0.633(+)	0	0.663(+)	0
F19	0.505(+)	0	0.375(+)	0	0.424(+)	0	0.223(+)	0	0.018(+)	0	0.350(+)	0
F20	0.252(+)	0	0.250(+)	0	0.250(+)	0	0.125(+)	0	0.005(+)	0	0.175(+)	0
最好个数	9		9		10		10		6		7	

(1)F1～F5 是 5 个较为简单的低维测试函数,全局最优解数量不超过 5 个,因此除 SDE 和 Self-CSDE 外,大多数算法均可稳定地找到所有解。

(2)虽然 F6～F10 也是低维函数,且问题最高维度不超过 3 维,但全局最优解数量却非常多,其中 F9 有多达 216 个全局最优解。从表 3 中可看出,除 CCDE-VN 和使用了多目标技术的 MOMMOP 外,其他算法均无法在这 5 个函数上稳定地找到所有解。以 F9 为例,除竞赛冠军算法 NMMSO 可偶尔找到所有解外,其他算法的 *PR* 值最高不超过 0.6,这表明 CCDE-VN 的竞争性拥挤小生境技术有很强的勘探能力,在有大量峰值的问题上能表现出更好性能。

(3)F11～F15 是低维的复杂函数,尽管全局最优解数量较少,最多不超过 8 个,但这 5 个函数均有较为崎岖的地形,对算法的开采能力有更高要求。从表 3 中可看出,F11 上 *PR* 值为 1 的对比算法仅有 4 个:FBK-DE、NCD-DE、AED-DDE 和 FDLS-ADE,而 F12 上 *PR* 为 1 的对比算法只有 AED-DDE 和 FDLS-ADE,F13 上 *PR* 值为 1 的对比算法仅有 FBK-DE。可看出 F11～F13 有较大的求解难度,大部分算法都无法取得较满意的结果。然而,本文提出的 CCDE-VN 在 F11～F13 上的 *PR* 值均为 1,即可稳定地找到全部最优解。在 F14 上,尽管 CCDE-VN 并非第一,但也仅差于 FBK-DE。在 F15 上,所有对比算法的 *PR* 值均未超过 0.73,而 CCDE-VN 却为 0.762,有较明显的优势。从这些分析可得出,CCDE-VN 在低维的复杂函数上同样优秀。值得一提的是,尽管 MOMMOP 在 F1～F10 上与 CCDE-VN 有相当性能,但其在 F11～F15 上却全部差于 CCDE-VN,这表明 MOMMOP 有较为优秀的勘探能力,但开采能力要差于 CCDE-VN。

(4)F16～F20 的维度要高于 F1～F15,求解难度也更大,其中维度最低为 5 维,最高为 20 维。在这 5 个函数中,CCDE-VN 在 F17 上明显好于 FBK-DE,在 F18 上与 FBK-DE 相当,而在 F16、F19、F20 上差于 FBK-DE。事实上,CCDE-VN 在 F16～F20 上的整体性能仅次于 FBK-DE,但明显优于其他 10 个对比算法。结合 CCDE-VN 在 F1～F15 上的整体性能,依然可得出 CCDE-VN 性能有很强的竞争力。

从上述实验结果和分析可看出,CCDE-VN 不仅有出色的勘探能力,也具备较强的开采能力,整体性能要优于对比算法。但需要注意的是,尽管 CCDE-VN 在大多数测试函数上要好于对比算法,但在 F14 和 F20 上与 FBK-DE 有较明显的差距。分析 CCDE-VN 在 F14 上的种群分布可发现,F14 的部分最优解所处的吸引盆较小,使得算法在第一阶段时就无法准确定位;而 F20 的吸引盆则过大,易导致算法的种群多样性过早下降,削弱算法在第一阶段的勘探能力。因此,对于存在吸引盆大小和形状分布不规律的函数,CCDE-VN 仍有提升空间,这也是我们后续研究工作的一个重要方向。

4.3　策略有效性验证

本小节将对本文提出的竞争性拥挤小生境技术、自适应高斯扰动策略以及虚拟小生境策略进行消融实验,以验证其有效性。

4.3.1　竞争性拥挤小生境技术验证

为验证竞争性拥挤技术的有效性,基于 CCDE-VN 设计了一个对比算法 CDE-VN。在 CDE-VN 中,用经典的拥挤技术替换竞争性拥挤技术,其他方面与 CCDE-VN 保持相同。为在统计意义上检验算法之间的性能是否存在显著性差异,本文采用非参数的 Wilcoxon 秩和检验,显著性水平设为 0.05,用符号“＋”表示 CCDE-VN 性能更好,“≈”表示 CCDE-VN 与对比算法性能相当,而“－”表示 CCDE-VN 性能更差。两个算法的实验结果如表 4 所示,从中可看出 CCDE-VN 在 5 个函数上要显著

表 4　竞争性拥挤小生境技术验证的实验结果

函数	CCDE-VN		CDE-VN	
	PR	*SR*	*PR*	*SR*
F1	**1**	1	**1**(≈)	1
F2	**1**	1	**1**(≈)	1
F3	**1**	1	**1**(≈)	1
F4	**1**	1	**1**(≈)	1
F5	**1**	1	**1**(≈)	1
F6	**1**	1	**1**(≈)	1
F7	**1**	1	0.863(＋)	0
F8	**1**	1	**1**(≈)	1
F9	**1**	1	0.433(＋)	0
F10	**1**	1	**1**(≈)	1
F11	**1**	1	**1**(≈)	1
F12	**1**	1	**1**(≈)	1
F13	**1**	1	**1**(≈)	1
F14	**0.846**	0.255	0.781(＋)	0.098
F15	**0.762**	0	0.723(＋)	0
F16	**0.680**	0	**0.673**(≈)	0
F17	**0.711**	0	0.679(＋)	0
F18	**0.667**	0	**0.667**(≈)	0
F19	**0.515**	0	**0.517**(≈)	0
F20	**0.265**	0	**0.265**(≈)	0
＋	/		5	
≈	/		15	
－	/		0	

地优于 CDE-VN，且未在任一函数上差于 CDE-VN。特别地，在 F7 和 F9 这类有大量全局最优解的问题上，CCDE-VN 性能提升明显，F7 的 *PR* 值提升 0.1 以上，F9 的 *PR* 值提升了 0.5 以上。值得注意的是，F9 是 CEC2015 测试集中全局最优解最多的函数，有 216 个全局最优解，且解的分布十分不均匀。这表明相比于经典的拥挤技术，竞争性拥挤技术显著增强了算法的勘探能力，有效提升了算法定位全局最优解的能力。

4.3.2 自适应高斯扰动策略验证

为验证该策略的有效性，设计了两个对比算法：1)CCDE-VN_noG，删除了自适应高斯扰动策略，其他方面与 CCDE-VN 保持相同；2)CCDE-VN_R，将自适应高斯扰动策略中的标准差 σ_i 设置为随机取值，不再按个体做个性化设置，即将式(11)的计算方式修改为下式：

$$\sigma_i = lb + rand(0,1) \cdot (ub - lb) \tag{17}$$

表 5 给出了 CCDE-VN 与上述两个对比算法的实验结果。从表中可看出，CCDE-VN 在 8 个测试函数上要优于 CCDE-VN_noG。以 F8 为例，图 6 展示了 F8 在二维时的函数图像，可看出其全局峰值所处地形非常陡峭，要求算法有很强的开采能力才能找到该全局峰值。在该函数上，CCDE-VN 比

表 5 自适应高斯扰动策略验证的实验结果

函数	CCDE-VN		CCDE-VN_noG		CCDE-VN_R	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**(≈)	1	**1**(≈)	1
F2	**1**	1	**1**(≈)	1	**1**(≈)	1
F3	**1**	1	**1**(≈)	1	**1**(≈)	1
F4	**1**	1	**1**(≈)	1	**1**(≈)	1
F5	**1**	1	**1**(≈)	1	**1**(≈)	1
F6	**1**	1	**1**(≈)	1	**1**(≈)	1
F7	**1**	1	**1**(≈)	1	**1**(≈)	1
F8	**1**	1	0.447(+)	0	0.971(+)	0.137
F9	**1**	1	**1**(≈)	1	**1**(≈)	1
F10	**1**	1	**1**(≈)	1	**1**(≈)	1
F11	**1**	1	**1**(≈)	1	**1**(≈)	1
F12	**1**	1	**1**(≈)	1	**1**(≈)	1
F13	**1**	1	0.895(+)	0.431	0.980(+)	0.882
F14	**0.846**	0.255	0.719(+)	0.020	0.755(+)	0
F15	**0.762**	0	0.730(+)	0	**0.755**(≈)	0
F16	**0.680**	0	0.667(+)	0	0.667(+)	0
F17	**0.711**	0	0.578(+)	0	**0.694**(≈)	0
F18	**0.667**	0	**0.657**(≈)	0	**0.667**(≈)	0
F19	0.515	0	0.189(+)	0	0.488(+)	0
F20	**0.265**	0	0.145(+)	0	0.196(+)	0
+	/		8		6	
≈	/		12		14	
−	/		0		0	

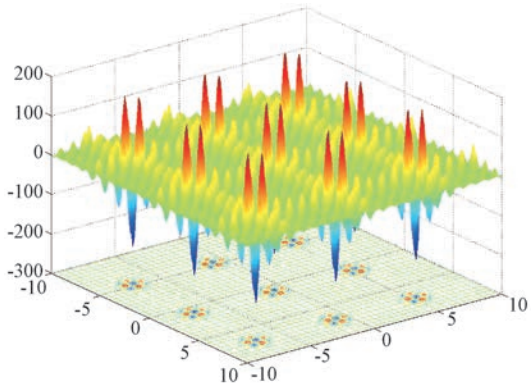


图 6 函数 F8 在二维时的图像

CCDE-VN_noG 的 *PR* 值高 0.5 以上，这说明自适应高斯扰动策略能显著提升算法的开采能力。与 CCDE-VN_R 相比，CCDE-VN 有 6 个测试函数更优，且未在任一个函数上差于 CCDE-VN_R，这验证了设置高斯扰动标准差时，同时考虑个体差异性与进化过程要优于仅考虑进化过程。

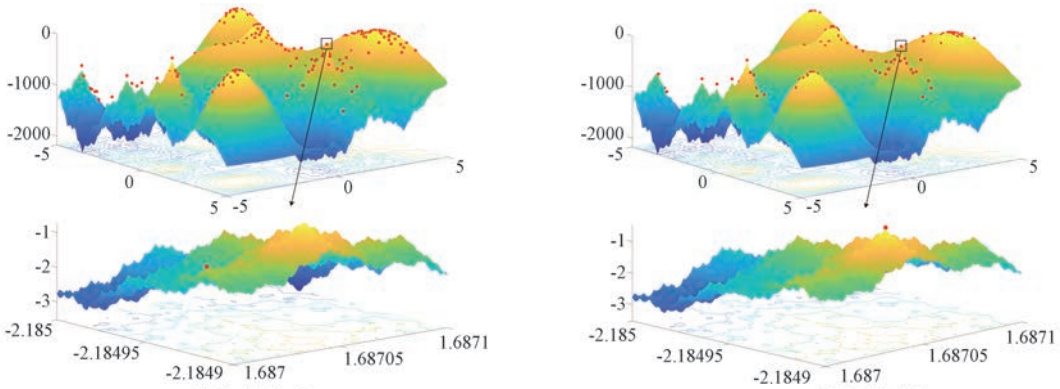
4.3.3 虚拟小生境策略验证

为搜索处于崎岖地形处的全局峰值，在 CCDE-VN 中采用了虚拟小生境策略。为验证该策略的有效性，设计了对比算法 CCDE，不再使用虚拟小生境策略，其他方面与 CCDE-VN 保持相同。CCDE-VN 与 CCDE 的对比结果如表 6 所示。

表 6 虚拟小生境策略验证的实验结果

函数	CCDE-VN		CCDE	
	PR	*SR*	*PR*	*SR*
F1	**1**	1	**1**(≈)	1
F2	**1**	1	**1**(≈)	1
F3	**1**	1	**1**(≈)	1
F4	**1**	1	**1**(≈)	1
F5	**1**	1	**1**(≈)	1
F6	**1**	1	**1**(≈)	1
F7	**1**	1	**1**(≈)	1
F8	**1**	1	**1**(≈)	1
F9	**1**	1	**1**(≈)	1
F10	**1**	1	**1**(≈)	1
F11	**1**	1	0.667(+)	0
F12	**1**	1	0.750(+)	0
F13	**1**	1	0.667(+)	0
F14	**0.846**	0.255	0.667(+)	0
F15	**0.762**	0	0.669(+)	0
F16	**0.680**	0	0.667(+)	0
F17	**0.711**	0	0.529(+)	0
F18	**0.667**	0	0.523(+)	0
F19	**0.515**	0	0.105(+)	0
F20	**0.265**	0	0(+)	0
+	/		10	
≈	/		10	
−	/		0	

从表 6 中可看出,CCDE-VN 在 F1-F10 这 10 个函数上的表现与 CCDE 相当,这说明两个算法在处理低维、地形平滑的测试函数时,性能差距不明显。然而,在 F11-F20 这 10 个复杂函数上,因其包含了较为崎岖的地形,使得 CCDE-VN 要显著优于 CCDE,这验证了虚拟小生境策略是有效的。为直观说明,图 7 给出了 CCDE 和 CCDE-VN 在 F11 上的种群分布情况,图中的红点代表个体。从两个子图的局部放大图可看出,全局峰值所处地形非常崎岖,包含大量局部峰值。尽管 CCDE 能搜索到全局峰值的周围空间,但因崎岖地形而陷入了局部峰值。相比之下,CCDE-VN 因采用了虚拟小生境策略,可更准确地定位全局峰值,避免局部峰值的影响。



(a) CCDE的种群分布情况　　(b) CCDE-VN的种群分布情况

图 7　CCDE 与 CCDE-VN 在 F11 上的种群分布情况对比

4.4　参数敏感性分析

4.4.1　阶段划分参数 η 敏感性分析

在 CCDE-VN 中,采用阶段划分参数 η 用于调节两个阶段的评估次数分配,第一阶段用于执行竞争性拥挤技术和自适应高斯扰动策略,而第二阶段执行虚拟小生境策略。通常,η 值越大,分配给第一阶段的评估次数更多,而第二阶段的评估次数会相对减少,此时算法勘探能力更强,但对于包含崎岖地形的优化问题易陷入局部峰值;相反,η 值越小,第二阶段的评估次数会增加,此时算法的开采能力更强,但勘探能力会减弱,易遗漏部分全局最优解。因此,有必要对 η 进行敏感性分析,以确定最优取值,更好地平衡勘探和开采能力。为此,本节实验中选取了五种不同的 η 值:$\eta=0.2$、0.4、0.6、0.8 和 0.9,实验结果如表 7 所示。从表中可看出,$\eta=0.8$ 时算法效果最佳,$\eta=0.6$ 略差于 $\eta=0.8$。对 $\eta=0.2$ 和 0.4,

表 7　不同 η 值的实验结果

函数	η=0.2		η=0.4		η=0.6		η=0.8		η=0.9	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F7	0.969	0.294	0.997	0.902	**1**	1	**1**	1	**1**	1
F8	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F9	0.993	0.569	**1**	1	**1**	1	**1**	1	**1**	1
F10	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F11	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F12	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F13	0.977	0.863	**1**	1	**1**	1	**1**	1	**1**	1
F14	0.670	0	0.670	0	0.830	0.176	**0.846**	0.255	0.820	0.157
F15	0.674	0	0.716	0	0.745	0	**0.762**	0	0.760	0
F16	0.667	0	0.667	0	0.667	0	**0.680**	0	0.670	0
F17	0.556	0	0.637	0	0.699	0	**0.711**	0	0.703	0
F18	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	0.316	0	0.380	0	**0.515**	0	**0.515**	0	0.375	0
F20	0.125	0	0.162	0	**0.275**	0	0.265	0	0.125	0
最好个数	11		13		16		**19**		14	

因第一阶段的评估次数不足，使得算法对最优解的定位能力也更差。而 $\eta=0.9$ 则将过多评估次数分配给第一阶段，导致算法对崎岖地形的处理能力不足，在更为复杂的 F19 和 F20 上出现明显的性能下降。综上，本文最终选取 $\eta=0.8$。

4.4.2 阈值 φ 敏感性分析

在竞争性拥挤小生境技术中，若父代种群中最相似的两个个体中“较差”的一个连续 φ 次未被子代个体成功替换，则会直接用子代个体替换该“较差”父代个体。若 φ 值太大，可能会使得子代个体长时间无法替换该“较差”父代个体，只能替换与其最相似的父代个体，增加了替换错误发生的可能性。相反，若 φ 值太小，则易使得直接替换的次数过多；同时，当父代种群中最相似的两个个体的相似度较低时，这也会进一步导致替换错误发生。因此，本节实验测试了五种 φ 值：$\varphi=10\cdot D$、$20\cdot D$、$30\cdot D$、$40\cdot D$、$50\cdot D$，实验结果如表 8 所示。从表中可看出，$\varphi=10\cdot D$ 在 F8 和 F9 上无法找到所有全局最优解，$\varphi=20\cdot D$ 在 F9 上表现不稳定，$\varphi=50\cdot D$ 则在 F7 和 F13 上表现不稳定。当 $\varphi=40\cdot D$ 时，算法可在 16 个函数上取得最好结果，但与 $\varphi=30\cdot D$ 相比，其性能差距并不明显。并且，$\varphi=30\cdot D$ 在 F14－F17 上均好于 $\varphi=40\cdot D$，仅在 F19 和 F20 差于 $\varphi=40\cdot D$。因此，综合来看 $\varphi=30\cdot D$ 最为合适。

表 8 不同 φ 值的实验结果

函数	$\varphi=10\cdot D$		$\varphi=20\cdot D$		$\varphi=30\cdot D$		$\varphi=40\cdot D$		$\varphi=50\cdot D$	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F7	**1**	1	**1**	1	**1**	1	**1**	1	0.999	0.980
F8	0.761	0	**1**	1	**1**	1	**1**	1	**1**	1
F9	0.780	0	0.985	0.275	**1**	1	**1**	1	**1**	1
F10	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F11	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F12	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F13	**1**	1	**1**	1	**1**	1	**1**	1	0.997	0.980
F14	**0.925**	0.588	0.886	0.373	0.846	0.255	0.814	0.118	0.814	0.196
F15	0.755	0	**0.767**	0	0.762	0	0.755	0	0.752	0
F16	**0.696**	0	0.683	0	0.680	0	0.673	0	0.670	0
F17	0.672	0	0.706	0	**0.711**	0	0.696	0	0.701	0
F18	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	0.500	0	0.507	0	0.515	0	**0.520**	0	0.517	0
F20	0.243	0	0.270	0	0.265	0	**0.277**	0	**0.277**	0
最好个数	14		14		15		**16**		13	

4.4.3 辅助个体数 k 敏感性分析

在虚拟小生境策略中，对崎岖地形处峰值的附近个体会生成 $k-1$ 个辅助个体来形成虚拟小生境。若 k 设置过小，虚拟小生境易过早收敛，难以帮助这些附近个体跳出局部峰值；若 k 值过大，又会使虚拟小生境占用过多评估次数，浪费算法的计算资源。因此，本小节对 k 值进行实验和分析，选取了与问题维度相关的四种取值：$k=5\cdot D$、$10\cdot D$、$15\cdot D$、$20\cdot D$，以及四种固定取值：$k=10$、30、60、100。值得注意的是，与问题维度相关的 k 值上限为 60。因此，共有八种不同的 k 值进行实验，对比结果如表 9 所示。可看出 $k=5\cdot D$、10、30 时，辅助个体数量较少，虚拟小生境易过早收敛，导致算法性能较差。$k=60$ 在 F13 上无法稳定地找到所有解。尽管 $k=100$ 可在 16 个函数上表现最好，但因虚拟小生境占用过多评估次数，导致在 F19 和 F20 上性能出现明显下降。当 $k=10\cdot D$、$15\cdot D$、$20\cdot D$ 时，算法性能最好，且彼此无明显区别，因此本文选择了这三者的中间值 $k=15\cdot D$。

4.4.4 DE 中 F 和 CR 敏感性分析

在算法 1 竞争性拥挤小生境技术和算法 4 虚拟小生境策略中，均采用 DE 作为搜索算法。对 DE 而言，F 和 CR 是其重要的两个控制参数。F 表示缩放因子，值越大，差分向量的扰动作用越大。CR

表 9　不同 k 值的实验结果

函数	$k=5\cdot D$		$k=10\cdot D$		$k=15\cdot D$		$k=20\cdot D$	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	**1**	1	**1**	1	**1**	1
F7	**1**	1	**1**	1	**1**	1	**1**	1
F8	**1**	1	**1**	1	**1**	1	**1**	1
F9	**1**	1	**1**	1	**1**	1	**1**	1
F10	**1**	1	**1**	1	**1**	1	**1**	1
F11	**1**	1	**1**	1	**1**	1	**1**	1
F12	**1**	1	**1**	1	**1**	1	**1**	1
F13	0.993	0.961	**1**	1	**1**	1	**1**	1
F14	0.846	0.275	0.846	0.255	0.846	0.255	0.830	0.216
F15	0.684	0	0.745	0	0.762	0	0.760	0
F16	0.673	0	**0.690**	0	0.680	0	0.667	0
F17	0.652	0	0.708	0	0.711	0	0.706	0
F18	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	0.512	0	**0.515**	0	**0.515**	0	**0.515**	0
F20	0.279	0	0.282	0	0.265	0	0.272	0
最好个数	13		**16**		15		15	
函数	$k=10$		$k=30$		$k=60$		$k=100$	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	**1**	1	**1**	1	**1**	1
F7	**1**	1	**1**	1	**1**	1	**1**	1
F8	**1**	1	**1**	1	**1**	1	**1**	1
F9	**1**	1	**1**	1	**1**	1	**1**	1
F10	**1**	1	**1**	1	**1**	1	**1**	1
F11	0.997	0.980	**1**	1	**1**	1	**1**	1
F12	**1**	1	**1**	1	**1**	1	**1**	1
F13	0.971	0.843	**1**	1	0.997	0.980	**1**	1
F14	0.817	0.157	0.859	0.314	**0.866**	0.314	0.846	0.216
F15	0.676	0	0.735	0	0.765	0	**0.772**	0
F16	0.667	0	0.676	0	0.667	0	0.673	0
F17	0.596	0	0.664	0	0.708	0	**0.730**	0
F18	0.552	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	0.157	0	0.495	0	0.512	0	0.468	0
F20	0	0	0.252	0	**0.287**	0	0.125	0
最好个数	12		14		15		**16**	

为交叉率,值越大,试验个体与变异个体越相似。在算法 1 中,[$F=0.1$,$CR=0.3$];而算法 4 中[$F=0.5$,$CR=0.9$]。因此,本小节对这两个算法的 F 和 CR 取值进行敏感性分析,分别对比四组不同取值。第 1 组和第 2 组是互换算法 1 和算法 4 的取值。第 3 组和第 4 组则分别采用了 DE 中的常用设置:[$F=0.5$,$CR=0.5$]、[$F=0.9$,$CR=0.1$]。表 10 和表 11 分别给出了对算法 1 和算法 4 的参数敏感性实验结果。

对算法 1 而言,从表 10 的结果可看出[$F=0.1$,$CR=0.3$]在 17 个函数上得到了最好结果。[$F=0.5$,$CR=0.9$]则效果较差,仅取得 13 个最好结果,尽管其在 F14 上取得较好结果,但在 F6-F9 上均无法稳定地找到所有解。[$F=0.5$,$CR=0.5$]

表 10 算法 1 中不同 F 与 CR 值的实验结果

函数	[$F=0.1$,$CR=0.3$]		[$F=0.5$,$CR=0.9$]		[$F=0.5$,$CR=0.5$]		[$F=0.9$,$CR=0.1$]	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	0.947	0.529	**1**	1	**1**	1
F7	**1**	1	0.991	0.745	0.999	0.980	**1**	1
F8	**1**	1	0.334	0	0.619	0	**1**	1
F9	**1**	1	0.571	0	0.826	0	0.988	0.392
F10	**1**	1	**1**	1	**1**	1	**1**	1
F11	**1**	1	**1**	1	**1**	1	**1**	1
F12	**1**	1	**1**	1	**1**	1	**1**	1
F13	**1**	1	**1**	1	**1**	1	**1**	1
F14	0.846	0.255	**0.987**	0.922	0.889	0.431	0.882	0.333
F15	**0.762**	0	**0.762**	0.020	0.750	0	0.752	0
F16	**0.680**	0	0.670	0	0.673	0	0.673	0
F17	0.711	0	**0.713**	0	0.706	0	0.708	0
F18	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	**0.515**	0	0.495	0	0.495	0	0.493	0
F20	0.265	0	0.223	0	0.252	0	**0.316**	0
最好个数	**17**		13		11		14	

表 11 算法 4 中不同 F 与 CR 值的实验结果

函数	[$F=0.5$,$CR=0.9$]		[$F=0.1$,$CR=0.3$]		[$F=0.5$,$CR=0.5$]		[$F=0.9$,$CR=0.1$]	
	PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	**1**	1	**1**	1	**1**	1	**1**	1
F2	**1**	1	**1**	1	**1**	1	**1**	1
F3	**1**	1	**1**	1	**1**	1	**1**	1
F4	**1**	1	**1**	1	**1**	1	**1**	1
F5	**1**	1	**1**	1	**1**	1	**1**	1
F6	**1**	1	**1**	1	**1**	1	**1**	1
F7	**1**	1	**1**	1	**1**	1	**1**	1
F8	**1**	1	**1**	1	**1**	1	**1**	1
F9	**1**	1	**1**	1	**1**	1	**1**	1
F10	**1**	1	**1**	1	**1**	1	**1**	1
F11	**1**	1	0.990	0.941	**1**	1	**1**	1
F12	**1**	1	0.988	0.902	**1**	1	**1**	1
F13	**1**	1	0.925	0.608	**1**	1	**1**	1
F14	**0.846**	0.255	0.843	0.235	0.840	0.235	0.667	0
F15	**0.762**	0	0.706	0	0.755	0	0.664	0
F16	**0.680**	0	0.676	0	0.673	0	0.667	0
F17	**0.711**	0	0.652	0	0.625	0	0.532	0
F18	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	0.515	0	0.426	0	**0.532**	0	0.164	0
F20	**0.265**	0	0.056	0	0.250	0	0	0
最好个数	**19**		11		15		14	

只取得 11 个最好结果，且在 F7～F9 上的结果不稳定。虽然[$F=0.9$,$CR=0.1$]在 F20 上表现较好，但在 F9 上表现不稳定，在 F15～F17 和 F19 上表现也略差于[$F=0.1$,$CR=0.3$]。综上考虑，采用[$F=0.1$,$CR=0.3$]为算法 1 的取值。

对算法 4 而言，从表 11 可看出[$F=0.5$,$CR=0.9$]有较明显优势，可在 19 个函数上取得最好结果。[$F=0.1$,$CR=0.3$]则相对较差，仅取得了 11 个最优，且在 F11～F13 上结果不稳定。[$F=0.5$,$CR=0.5$]的整体性能略差于[$F=0.5$,$CR=0.9$]，

且在 F17 上有较为明显的性能下降。[$F=0.9$,$CR=0.1$]在 F14～F17 和 F19～F20 上表现较差。因此,最终选择[$F=0.5$,$CR=0.9$]作为算法 4 的取值。

4.4.5　种群规模 NP 敏感性分析

为分析种群规模 NP 对算法性能的影响,本小节测试五种代表性取值:$NP=50$、100、200、300、400,表 12 给出了实验对比结果。从表中可得到三点结论:(1) NP 值应大于最优解数量,否则会因个体数量不足而遗漏部分最优解。例如,F9 有 216 个最优解,当 $NP=50$、100、200 均无法找到所有最优解。(2)对 F1-F5 这类较为简单的低维测试函数,CCDE-VN 对种群规模不敏感,NP 略大于或小于推荐值均能找到所有最优解。(3)对 F16-F20 这类维度较高的复杂函数,NP 较大或较小都不适宜。一方面,虽然较大 NP 可增强勘探能力,有助于定位更多最优解,但也会因有限的适应度函数评估次数而降低开采能力,导致算法的收敛精度不够。另一方面,较小 NP 有利于提高开采能力,但会因勘探能力不足而导致遗漏部分最优解。综上所述,对于不同类型的问题,种群规模设置也应有所区别。

表 12　不同 *NP* 值的实验结果

函数	推荐 NP	最优解数量	$NP=50$		$NP=100$		$NP=200$		$NP=300$		$NP=400$	
			PR	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*	*PR*	*SR*
F1	80	2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F2	80	5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F3	80	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F4	80	4	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F5	80	2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F6	100	18	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F7	300	36	0.999	0.980	**1**	1	**1**	1	**1**	1	**1**	1
F8	300	81	0.440	0	0.932	0.020	**1**	1	**1**	1	**1**	1
F9	300	216	0.231	0	0.450	0	0.804	0	**1**	1	**1**	1
F10	100	12	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F11	200	6	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
F12	200	8	0.968	0.745	**1**	1	**1**	1	**1**	1	**1**	1
F13	200	6	**1**	1	**1**	1	**1**	1	**1**	1	0.990	0.941
F14	200	6	0.752	0.039	0.768	0.039	0.856	0.314	**0.886**	0.373	0.882	0.392
F15	200	8	0.748	0	0.755	0	**0.762**	0.020	0.745	0	0.750	0
F16	200	6	0.676	0	0.683	0	**0.690**	0	0.670	0	0.683	0
F17	200	8	0.672	0	0.696	0	**0.708**	0	**0.708**	0	0.691	0
F18	200	6	0.631	0	**0.667**	0	**0.667**	0	**0.667**	0	**0.667**	0
F19	200	8	0.449	0	0.502	0	**0.512**	0	0.507	0	0.387	0
F20	200	8	0.294	0	**0.297**	0	0.272	0	0.154	0	0.127	0

4.5　在实际优化问题上的应用

4.5.1　三角函数超越方程组

在实际工程领域中,很多优化问题涉及三角函数超越方程组的求解[38],例如电机结构设计[39]、传热问题[40]以及多杆复杂机构位置分析问题[41]等。这些问题中的三角函数超越方程组通常包含多个最优解,本质上是一个多峰优化问题,求解难度较大。式(18)给出了一个三角函数超越方程组的实例[38],该实例有 18 个全局最优解。

$$\begin{cases} F_1(X)=1+\sin(x_2-x_1)\cdot\left[\left(x_1-\dfrac{\pi}{4}\right)\cdot\dfrac{\sin x_1}{\sin x_2}+\left(x_2-\dfrac{3\pi}{4}\right)\cdot\dfrac{\sin x_2}{\sin x_1}\right]=0 \\ F_2(X)=1-\sin(x_2-x_1)\cdot\left[\left(x_1-\dfrac{\pi}{4}\right)\cdot\dfrac{\cos x_1}{\sin x_2}+\left(x_2-\dfrac{3\pi}{4}\right)\cdot\dfrac{\cos x_2}{\sin x_1}\right]=0 \end{cases} \tag{18}$$

本文采用 CCDE-VN 对其进行求解,与 CEC2015 多峰优化竞赛的冠军算法 NMMSO[37]、2023 年提出的 FDLS-ADE 算法[32]以及在实际问题中有较好表现的两个多峰优化算法——HNDE/2A[42]和 KSDE[43]进行对比。因该问题是求零点解,不宜直接用优化算法求解,可先转化为求最大值或最小值问题,例如:求所有方程的绝对值之和或平方和的最小值问题,可形式化为 $\min\sum_{i=1}^{m}|e_i(x)|$ 或 $\min\sum_{i=1}^{m}e_i^{\,2}(x)$,其中 $e_i(x)$ 表示方程组中的第 i 个方程,m 是方程个数。方便起见,本文采用求平方和的最小值,即 $\min\sum_{i=1}^{m}e_i^{\,2}(x)$。实验中每个算法运行 51 次,评估次数设为 200000。除 NMMSO 采用了动态种群外,其他算法的种群规模均设为 100。算法的 *PR* 值和 *SR* 值分别如图 8 与图 9 所示。

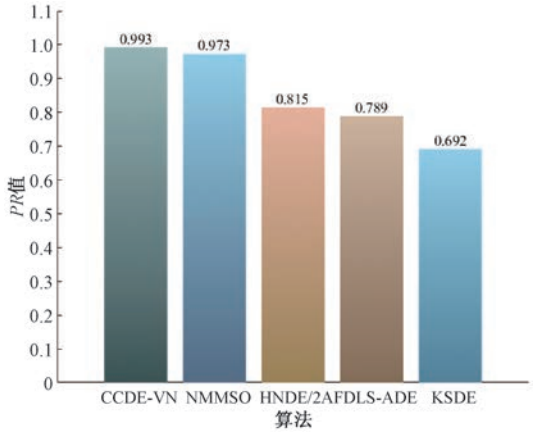


图 8 算法在三角函数超越方程组问题上的 *PR* 值

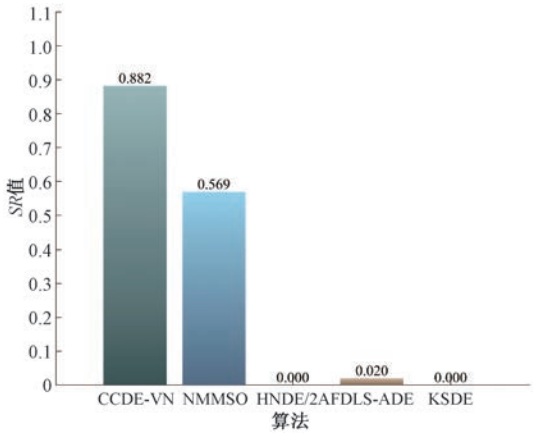


图 9 算法在三角函数超越方程组问题上的 *SR* 值

从这两个图中可看出，CCDE-VN 的 *PR* 值和 *SR* 值要优于其他对比算法。尽管 CCDE-VN 与 NMMSO 在 *PR* 值上差距不大，但在 *SR* 值上 CCDE-VN 比 NMMSO 高 0.3 以上，即 CCDE-VN 在 51 次运行中找到所有解的次数要显著高于 NMMSO。该实验表明 CCDE-VN 有较好的鲁棒性，在求解难度较大的实际问题上也能表现出优异性能。

4.5.2 调频声波合成问题

调频声波合成[44]是音频信号处理中的一个重要问题，通过调节波形生成器的相关参数，可调频合成复杂且多样的音色，在音乐合成、声音设计以及数字音频处理等领域有广泛应用。该问题的求解目标是按给定的频率特性和音色生成声波信号，可形式化为式(19)－(20)[44]：

$$y(t)=a_1\sin(w_1t\theta+a_2\sin(w_2t\theta+a_3\sin(w_3t\theta))) \tag{19}$$

$$y_0(t)=1.0\sin(0.5t\theta+1.5\sin(4.8t\theta+2.0\sin(4.9t\theta))) \tag{20}$$

其中 $\theta=2\pi/100$，其他参数的取值范围是$[-6.4,6.35]$。求解该问题的关键是找到最优解 $X=\{a_1,w_1,a_2,w_2,a_3,w_3\}$，使得由式(19)生成的估计声波与式(20)给定的目标声波尽可能相似。因此，可通过最小化估计声波与目标声波之间的误差平方和来求解，如式(21)所示：

$$f(x)=\sum_{t=0}^{100}(y(t)-y_0(t))^2 \tag{21}$$

该问题是一个高度非线性且存在大量局部峰值的 MMOP，最优解为 $f(x)=0$，有 8 个已知的全局最优解。

本文用 CCDE-VN 求解该问题，并与 CEC2015 多峰竞赛的冠军算法 NMMSO[37]、FDLS-ADE 算法[32]以及两个性能较好的其他算法——DIDE[19]和 DHNDE[15]进行对比。每个算法运行 51 次，适应度函数评估次数为 200000。除 NMMSO 采用动态种群外，其他算法的种群规模均设为 100。算法的 *PR* 值如图 10 所示。因该问题非常复杂，所有算法均无法在给定的评估次数内找到所有解，即 *SR* 全为 0，因此未给出对应的 *SR* 图。从图 10 中可看出，CCDE-VN 的 *PR* 值为 0.336，排名第一，而其他对比算法的 *PR* 值最高仅为 0.015。因此，这也进一步验证了 CCDE-VN 性能非常有竞争力。

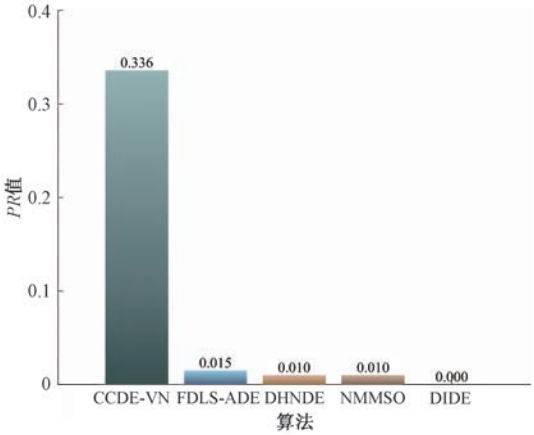


图 10 算法在调频声波合成问题上的 *PR* 值

4.5.3 求解实际问题的时间对比情况

为验证 CCDE-VN 求解上述两个实际问题的运行时间情况，本小节对比了 CCDE-VN 和其他对比算法的实际耗时情况，分别在表 13 和 14 中给出具体数据。可看出，虽然 CCDE-VN 在两个实际问题上的平均运行时间并非最短，但也能排至第二。总体来看，CCDE-VN 不论是在求解精度，还是在运行时间上均有较好表现。

表 13　各算法求解三角函数超越方程组的时间

算法	时间/s
FDLS-ADE	**3.94**
CCDE-VN	4.39
KSDE	9.48
NMMSO	17.65
HAND/2A	18.63

表 14　各算法求解调频声波合成问题的时间

算法	时间/s
DIDE	**6.37**
CCDE-VN	7.12
FDLS-ADE	8.05
DHNDE	11.37
NMMSO	17.65

5　结　论

经典的拥挤小生境技术是求解多峰优化问题的一种有效手段,但其存在替换错误问题,易导致部分全局最优解无法被定位。此外,在多峰优化问题中,有些全局峰值所处地形较为崎岖,使得算法易陷入局部最优,对问题求解带来很大挑战。为此,本文提出了一种基于竞争性拥挤小生境技术的多峰优化算法,简称 CCDE-VN,主要贡献有两点:(1)提出了竞争性拥挤小生境技术,不仅考虑子代个体与父代个体之间的相似性,同时还考虑了父代个体自身之间的相似性,可减少替换错误的发生,更好地保持种群多样性。(2)提出了虚拟小生境技术,在可能处于局部峰值的个体周围生成若干辅助个体,形成一个虚拟小生境,以进一步对该个体做细粒度搜索,从而更好地定位到全局峰值。

在 CEC2015 测试集和两个实际优化问题上进行了大量实验验证,可得到如下六点结论:(1)在竞争性拥挤技术中,通过增加可替换个体的方式,能有效减少替换错误的发生;(2)在设置高斯扰动的标准差时,同时考虑进化过程与个体差异性这两点因素有更好效果;(3)对处于崎岖地形处的全局峰值,有必要进一步处理其附近个体,以提高定位到全局峰值的可能性;(4)本文提出的三种策略之间能实现优势互补;(5)与其他 11 种有代表性的多峰优化算法的对比结果表明,CCDE-VN 性能有很强的竞争力;(6)CCDE-VN 在实际优化问题上也有很好性能。在未来,将对竞争性拥挤小生境技术做进一步研究:(1)尝试去除参数 φ,使得该技术有更好普适性,能作为一个通用框架用于改善其他同类算法性能;(2)进一步提高在高维函数上的性能,例如:F16-F20;(3)尝试用于求解更多的实际优化问题,例如:桁架结构优化。

致　谢　感谢西安电子科技大学高卫峰教授对本文提出的中肯意见,使得本文的质量得以提升!感谢九江学院彭虎教授在算法对比实验部分提供的帮助!

参　考　文　献

[1] Wang Z, Chen D, Gong J, et al. Fast high-precision ellipse detection method. Pattern Recognition, 2021, 111: 107741

[2] AlQuraishi M. Machine learning in protein structure prediction. Current Opinion in Chemical Biology, 2021, 65: 1-8

[3] Jafari M , Salajegheh E , Salajegheh J. Optimal design of truss structures using a hybrid method based on particle swarm optimizer and cultural algorithm. Structures, 2021, 32:391-405

[4] Hu X M, Zhang S R, Li M, et al. Multimodal particle swarm optimization for feature selection. Applied Soft Computing, 2021, 113: 107887

[5] Zhao H, Li J R, Liu J. Localized time-distance-based multimodal optimization algorithm and its application in PID control. Chinese Journal of Computers, 2024, 47 (6): 1323-1340 (in Chinese)
(赵宏,李珈瑞,刘静. 基于局部时空的多峰优化算法及其在PID控制中的应用. 计算机学报, 2024, 47 (6): 1323-1340)

[6] Cheng Z G, Zhan Z H. Two-layer collaborative differential evolution algorithm for multimodal optimization problems. Chinese Journal of Computers, 2021, 44(9): 1806-1823 (in Chinese)
(陈宗淦,詹志辉. 面向多峰优化问题的双层协同差分进化算法. 计算机学报, 2021, 44(9): 1806-1823)

[7] Li M, Zhao Y, Cao R, et al. A recursive framework for improving the performance of multi-objective differential evolution algorithms for gene selection. Swarm and Evolutionary Computation, 2024, 87: 101546

[8] Cavallaro C, Cutello V, Pavone M, et al. Machine learning and genetic algorithms: a case study on image reconstruction. Knowledge-Based Systems, 2024, 284: 111194

[9] Zhou X Y, Yin Z Y, Gao W F, Tan G S, Yi Y G. Adaptive multi-neighborhood artificial bee colony algorithm based on reinforcement learning. Chinese Journal of Computers, 2024, 47 (7): 203-228 (in Chinese)
(周新宇,尹子悦,高卫峰,谭贵森,易玉根. 一种基于强化学习的自适应多邻域人工蜂群算法. 计算机学报, 2024, 47(7): 203-228)

[10] Tijjani S, Ab Wahab M N, Noor M H M. An enhanced particle swarm optimization with position update for optimal feature selection. Expert Systems with Applications, 2024, 247: 12-37

[11] Thomsen R. Multimodal optimization using crowding-based differential evolution//Proceedings of the IEEE Congress on Evolutionary Computation. Portland, USA, 2004: 1382-1389

[12] Li X. Efficient differential evolution using speciation for multimodal function optimization//Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. New York, USA, 2005: 873-880

[13] Li X. A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio//Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. London, UK, 2007: 78-85

[14] Gao W, Yen G G, Liu S. A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. IEEE Transactions on Cybernetics, 2013, 44(8): 1314-1327

[15] Wang K, Gong W, Deng L, et al. Multimodal optimization via dynamically hybrid niching differential evolution. Knowledge-Based Systems, 2022, 238: 107972

[16] Zhao H, Zhan Z H, Liu J. Outlier aware differential evolution for multimodal optimization problems. Applied Soft Computing, 2023, 140: 110264

[17] Zhang Y H, Gong Y J, Gao Y, et al. Parameter-free Voronoi neighborhood for evolutionary multimodal optimization. IEEE Transactions on Evolutionary Computation, 2019, 24(2): 335-349

[18] Merz P, Freisleben B. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation, 2000,4(4):337-352

[19] Chen Z G, Zhan Z H, Wang H, et al. Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems. IEEE Transactions on Evolutionary Computation, 2019, 24(4): 708-719

[20] Zhou X Y, Huang J H, Peng H, et al. Differential evolution algorithm based on adaptive bi-coordinate systems for mixed-variable optimization problem. Chinese Journal of Computers, 2024, 47(09): 2116-2140 (in Chinese)
(周新宇,黄君洪,彭虎,等. 自适应双坐标系的差分进化算法求解混合变量优化问题. 计算机学报, 2024, 47(09): 2116-2140)

[21] Lin X, Luo W, Xu P. Differential evolution for multimodal optimization with species by nearest-better clustering. IEEE Transactions on Cybernetics, 2019, 51(2): 970-983

[22] Jiang Y, Zhan Z H, Tan K C, et al. Optimizing niche center for multimodal optimization problems. IEEE Transactions on Cybernetics, 2021, 53(4): 2544-2557

[23] Liang S M, Wang Z J, Huang Y B, et al. Niche center identification differential evolution for multimodal optimization problems. Information Sciences, 2024, 678: 121009

[24] Wang Z J, Zhou Y R, Zhang J. Adaptive estimation distribution distributed differential evolution for multimodal optimization problems. IEEE Transactions on Cybernetics, 2020, 52(7): 6059-6070

[25] Zhao H, Zhan Z H, Lin Y, et al. Local binary pattern-based adaptive differential evolution for multimodal optimization problems. IEEE Transactions on Cybernetics, 2019, 50(7): 3343-3357

[26] Zhou X, Li N, Fan L, et al. Adaptive niching differential evolution algorithm with landscape analysis for multimodal optimization. Information Sciences, 2025, 700: 121842

[27] Cheng R, Li M, Li K, et al. Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection. IEEE Transactions on Evolutionary Computation, 2017, 22(5): 692-706

[28] Basak A, Das S, Tan K C. Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection. IEEE Transactions on Evolutionary Computation, 2012, 17(5): 666-685

[29] Bandaru S, Deb K. A parameterless-niching-assisted bi-objective approach to multimodal optimization//Proceedings of the IEEE Congress on Evolutionary Computation. Cancun, Mexico, 2013: 95-102

[30] Wang Y, Li H X, Yen G G, et al. MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems. IEEE Transactions on Cybernetics, 2014, 45(4): 830-843

[31] Wang R, Hao K, Huang B, et al. Adaptive niching particle swarm optimization with local search for multimodal optimization. Applied Soft Computing, 2023, 133: 109923

[32] Wang Z J, Zhan Z H, Li Y, et al. Fitness and distance based local search with adaptive differential evolution for multimodal optimization problems. IEEE Transactions on Emerging Topics in Computational Intelligence, 2023, 7(3): 684-699

[33] Sheng W, Wang X, Wang Z, et al. Adaptive memetic differential evolution with niching competition and supporting archive strategies for multimodal optimization. Information Sciences, 2021, 573: 316-331

[34] Dick G, Whigham P A. Weighted local sharing and local clearing for multimodal optimisation. Soft Computing, 2011, 15: 1707-1721

[35] Li X, Engelbrecht A, Epitropakis M G. Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Technical Report, 2013

[36] Epitropakis M G, Li X, Burke E K. A dynamic archive niching differential evolution algorithm for multimodal optimization//Proceedings of the IEEE Congress on Evolutionary Computation. Cancun, Mexico, 2013: 79-86

[37] Fieldsend J E. Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser//Proceedings of the IEEE Congress on Evolutionary Computation. Beijing, China, 2014: 2593-2600

[38] Li T J, Jia J Y, Hu X M. Global set of solutions for two types of systems of nonlinear equations in mechanical engi-

neering. Journal of Xidian University, 2005, 32 (1): 71-74 (in Chinese)
(李团结,贾建援,胡雪梅. 机械工程中两类非线性方程组的完全解. 西安电子科技大学学报(自然科学版), 2005, 32 (1): 71-74)
[39] Kumar P, Hati A S. Review on machine learning algorithm based fault detection in induction motors. Archives of Computational Methods in Engineering, 2021, 28(3): 1929-1940
[40] Liu D, Cheng Y M. The interpolating element-free Galerkin method for three-dimensional transient heat conduction problems. Results in Physics, 2020, 19: 103477
[41] Kapsalyamov A, Hussain S, Brown N A T, et al. Synthesis of a six-bar mechanism for generating knee and ankle motion trajectories using deep generative neural network. Engineering Applications of Artificial Intelligence, 2023, 117: 105500
[42] Wang K, Gong W, Liao Z, et al. Hybrid niching-based differential evolution with two archives for nonlinear equation system. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(12): 7469-7481
[43] Wu J, Gong W, Wang L. A clustering-based differential evolution with different crowding factors for nonlinear equations system. Applied Soft Computing, 2021, 98: 106733
[44] Zhou X, Wu Z, Wang H, et al. Gaussian bare-bones artificial bee colony algorithm. Soft Computing, 2016, 20(3):907-924

ZHOU Xin-Yu, Ph. D., associate professor. His main research interests include intelligent computation.

TIAN Long-Hui, M. Eng. candidate. His main research interest is swarm intelligent optimization algorithms.

MING Fei, Ph. D. candidate. His main research interests include intelligent optimization and evolutionary multitask optimization.

GONG Wen-Yin, Ph. D., professor. His main research interests include intelligent computation and multi-objective optimization.

WANG Hui, Ph. D., professor. His main research interests include intelligent computation and multimodal optimization.

Background

In the real world, there exist many multimodal optimization problems, such as multi-ellipse detection, protein structure prediction, truss structure optimization, and feature selection. These problems are characterized by multiple global optima, requiring algorithms to find as many of these optima as possible. Although the evolutionary algorithm (EA) is an effective optimization approach suitable for unimodal optimization problems, it lacks mechanisms to maintain population diversity, and therefore, its effectiveness in solving multimodal optimization problems remains low. To address this, niching techniques have been developed to enhance the diversity maintenance capability of EA, thereby improving its effectiveness in solving multimodal optimization problems. Among these niching techniques, crowding niching has received widespread attention due to its simplicity and high effectiveness.

However, the classical crowding niching technique only considers the similarity between offspring and parents. If offspring individuals are dissimilar to their parents, this can lead to "replacement errors," which significantly impacts algorithm performance. To address this issue, this paper proposes an improved crowding niching technique. Unlike the classical crowding niching technique, which considers only the similarity between offspring and parents, our approach also considers the similarity between parents. By identifying the two most similar parents, the worse of the two in terms of fitness value is considered a replaceable individual, thereby reducing replacement errors. Extensive experiments were conducted on the CEC2015 benchmark set, and 11 representative algorithms were included in the comparison, including two championship algorithms from the CEC multimodal optimization competition. The comparison results demonstrate that our approach exhibits highly competitive performance.

This work was supported by the National Natural Science Foundation of China under Grant Nos. 62366022 and 62166027, the Outstanding Youth Project of Jiangxi Natural Science Foundation under Grant No. 20212ACB212004, and the Jiangxi Provincial Natural Science Foundation under Grant No. 20232BAB202048.